robotron

Anleitung für den
Programmierer

Numerische Bibliotheken

Arbeitsplatzcomputer A 7100 Betriebssystem SCP 1700

SYSTEMUNTERLAGEN-DOKUMENTATION 6/86	Numerische Bibliotheken Anleitung für den Programmierer	MOS
		SCP 1700

Anleitung für den Programmierer

Numerische Bibliotheken

AC A7100

VEB Robotron-Projekt Dresden

C 1014-0001-1 M 3030

Die vorliegende Systemunterlagendokumentation, Anleitung für den Programmierer Numerische Bibliotheken , entspricht dem Stand von 6/86.

Nachdruck, jegliche Vervielfältigung oder Auszüge daraus sind unzulässig.

Die Ausarbeitung erfolgte durch ein Kollektiv des VEB Robotron-Elektronik Dresden .

Herausgeber:

VEB Robotron-Projekt Dresden 8010 Dresden, Leningrader Str. 9

(C) 1986

Kurzreferat

Die Numerischen Bibliotheken bestehen aus Programmen für mathematische Grundoperationen (Emulator), für die Anpassung an den jeweiligen Prozessor (Interfacebibliotheken), für Konvertierungen, für mathematische Funktionen und Programmen zum Schreiben von Error-Handlern. Sie sind vor allem im Datenformat Gleitkomma realisiert.

Ag 706/311/87

Inhaltsverzeichnis | Seite

1.	Vorbemerkungen	7
2.	Prozessor	8
2.1.	Vorbemerkungen	8
2.2.	Zahlendarstellung	8
2.2.1.	Zahlensystem	8
2.2.2.	Datenformate	8
2.2.2.1.	Übersicht	8
2.2.2.2.	Codierung der Datenarten	9
2.2.2.3.	Integer-Zahlen	9
2.2.2.4.	BCD-Zahlen (packed decimal)	9
2.2.5.	Real-Zahlen (Gleitkommazahlen)	10
2.3.	Spezielle Werte	11
2.3.1.	Vorbemerkungen	11
2.3.2.	Nonnormals (nichtnormalisierte Zahlen)	11
2.3.3.	Denormals (denormalisierte Zahlen)	11
2.3.4.	Unnormals (Unnormalisierte Zahlen)	11
2.3.5.	Nullen und Pseudonullen	11
2.3.6.	Infinities	11
2.3.7.	NaNs	12
2.3.8.	Indefinities	12
2.4.	Befehlssatz der Prozessoren	12
2.5.	Register der Prozessoren	12
2.5.1.	Statuswort	12
2.5.2.	Steuerwort	13
2.5.3.	Register-Stack	14
2.6.	Exceptions der Prozessoren	15
2.6.1.	Exceptions und deren Ursachen	15
2.6.2.	Auswertung der Exceptions	16
2.6.3.	Exception-Handler	16
2.7.	Arbeit mit den Prozessoren	16
2.7.1.	Arbeit mit dem Emulator EWM87.OBJ	16
2.7.2.	Arbeit mit dem Schaltkreis WM87	17
3.	Interfacebibliotheken	18
4.	Konvertierungsbibliothek DCON87	19
4.1.	Vorbemerkungen	19
4.2.	Vereinbarungen im Assemblerprogramm	19
4.3.	Stacknutzung	20
4.4.	Genauigkeit der Konvertierungsalgorithmen	20
4.5.	Exceptions der DCON87	21
4.6.	Format der Dezimalzahlen bei mqcDEC_BIN	22
4.7.	Linken der Programme	23
4.8.	Beschreibung der Konvertierungsprogramme	23
4.8.1.	Konvertierung binär in dezimal mqcBIN_DECLOW	23
4.8.2.	Konvertierung dezimal in binär mqcDEC_BIN	25
4.8.3.	Konvertierung dezimal in binär mqcDECLOW_BIN	27
4.8.4.	Konvertierung long-real in temporary-real mqcLONG_TEMP	28
4.8.5.	Konvertierung short-real in temporary-real mqcSHORT_TEMP	29
4.8.6.	Konvertierung temporary-real in long-real mqcTEMP_LONG	30
4.8.7.	Konvertierung temporary-real in short-real mqcTEMP_SHORT	32

5.	Bibliothek für mathematische Funktionen	34
5.1.	Vorbemerkungen	34
5.2.	Vereinbarung der CEL87-Programme in ASM86-Programmen	35
5.3.	Stack-Nutzung	35
5.4.	Registernutzung	36
5.5.	Exceptions	36
5.6.	Linken der Programme	37
5.7.	Beschreibung der Programme	37
5.7.1.	Programm zur Berechnung des Arcuscosinus mqerACS	37
5.7.2.	Programm zur Berechnung des Arcussinus mqerASN	38
5.7.3.	Programm zum Berechnen des Arcustangens mqerAT2	39
5.7.4.	Programm zur Berechnung des Arcustangens mqerATN	42
5.7.5.	Programm zur Berechnung des Cosinus mqerCOS	43
5.7.6.	Programm zur Berechnung des Cosinus Hyperbolicus mqerCSH	44
5.7.7.	Programm zur Bildung der positiven Differenz mqerDIM	45
5.7.8.	Programm zur Berechnung einer Potenz mqerEXP	46
5.7.9.	Programm zum Runden real in word-integer mqerIA2	48
5.7.10.	Programm zum Runden real in short-integer mqerIA4	49
5.7.11.	Programm zum Runden real in integer mqerIAX	50
5.7.12.	Programm zur Bildung von word-integer aus real mqerIC2	51
5.7.13.	Programm zur Bildung von short-integer aus real mqerIC4	53
5.7.14.	Programm zur Abspaltung des gebrochenen Teils mqerICX	54
5.7.15.	Programm zum Runden real in word-integer mqerIE2	55
5.7.16.	Programm zum Runden real im short-integer mqerIE4	56
5.7.17.	Programm zum Runden real in integer mqerIEX	57
5.7.18.	Programm zur Berechnung des Dezimalexponenten mqerLGD	58
5.7.19.	Programm zur Berechnung des Logarithmus naturalis mqerLGE	59
5.7.20.	Programm zur Berechnung des Maximums mqerMAX	60
5.7.21.	Programm zur Berechnung des Minimums mqerMIN	62
5.7.22.	Programm zur Berechnung des Restes einer Division mqerMOD	62
5.7.23.	Programm zur Berechnung des Restes einer Division mqerRMD	64
5.7.24.	Programm zur Bildung von y mit dem Vorzeichen von x mqerSGN	65
5.7.25.	Programm zur Berechnung des Sinus mqerSIN	67
5.7.26.	Programm zur Berechnung des Sinus Hyperbolicus mqerSNH	68
5.7.27.	Programm zur Berechnung des Tangens mqerTAN	69
5.7.28.	Programm zur Berechnung des Tangens Hyperbolicus mqerTNH	70
5.7.29.	Programm zur Berechnung einer Potenz mqerY2X	71
5.7.30.	Programm zur Berechnung einer Potenz mqerYI2	73
5.7.31.	Programm zur Berechnung einer Potenz mqerYI4	74

5.7.32.	Programm zur Berechnung einer Potenz mqerYIS	76
6.	Exception-Handler-Bibliothek EH87	78
6.1.	Vorbemerkungen	78
6.2.	Begriffe	78
6.2.1.	Non-trapping-NaNs	78
6.2.2.	Non-ordered comparisons	79
6.3.	Arbeit mit den Programmen	79
6.3.1.	Datenfeld ESTATE87	79
6.3.2.	Regeln für die Nutzung der EH87	81
6.3.3.	Linken der EH87	82
6.4.	Beschreibung von Teilprogrammen	82
6.4.1.	DECODE	82
6.4.2.	ENCODE	83
6.4.3.	FILTER	84
6.4.4.	NORMAL	85
6.4.5.	SIEVE	86
Anlage 1	Datenarten des WM87	87
Anlage 2	Codierung in der Datenart integer	88
Anlage 3	Codierung in der Datenart BCD	89
Anlage 4	Codierung in den Datenarten short- und long-real	90
Anlage 5	Codierung in der Datenart temporary-real	92
Anlage 6	Befehle des Prozessors WM87	94
Anlage 7	Status des Prozessors nach Initialisierung durch INIT87	98
Anlage 8	Reaktion des Prozessors auf maskierte Exceptions	99
Anlage 9	Maskierte Reaktion bei Rundung mit Überlauf	103
Anlage 10	Reaktion der Prozessoren auf die maskierten und unmaskierten Exceptions	104
Anlage 11	Ergebnisse bei Infinity-Operanden	105
Anlage 12	Ergebnisse bei Null-Operanden	107
Anlage 13	Ergebnisse bei unnormalisierten Operanden (Unormals)	109
Anlage 14	Datenfeld ESTATE87 der EH87	110
Anlage 15	Globale Symbole	111
Anlage 16	Programme der CEL87.L86	114
Sachwortverzeichnis		115

1. Vorbemerkungen

Die Programme der numerischen Bibliotheken realisieren Operationen in den Datenformaten integer, BCD und Gleitkomma.
Kernstück dieser Bibliotheken ist der Prozessor WM87. Dabei kann es sich sowohl um den Schaltkreis WM87 als auch dessen softwaremäßigen Simulator (Emulator WM87) handeln. Beide realisieren den kompletten Befehlssatz gemäß Anlage 6 und sind anwenderspezifisch weitgehend identisch.
Die Interfacebibliotheken WM87.L86 und EWM87.L86 dienen zur Anpassung der aufrufenden Programme an den Schaltkreis bzw. den Emulator.
Die Konvertierungsbibliothek DCON87.L86 enthält Routinen zur Umformung von Zahlen in externer Darstellung in die rechnerinterne Darstellung und umgekehrt. Für die Berechnung verschiedener mathematischer Funktionen stehen die Programme der CEL87.L86 zur Verfügung. Als Hilfsmittel beim Schreiben von Exception-Handlern dienen die Prozeduren der EH87.L86.

2. Prozessor

2.1. Vorbemerkungen

Unter dem Prozessor wird im folgenden sowohl der Schaltkreis WM87 als auch dessen softwaremäßiger Simulator EWM87.OBJ (Emulator) verstanden. Anwenderspezifisch bestehen keine prinzipiellen Unterschiede.
Der WM87-Emulator ist ein Programmpaket, das die Register und Funktionen des WM87 auf der Basis des WM86 softwaremäßig nachbildet. Er kann genutzt werden, wenn der Schaltkreis nicht zur Verfügung steht oder die höhere Operationsgeschwindigkeit nicht benötigt wird.
Die Verbindung zwischen den Prozessoren (WM87 oder Emulator) und dem aufrufenden Programm wird durch die Interfacebibliotheken EWM87.L86, WM87.L86 oder 87NULL.L86 hergestellt (siehe Abschnitt 3.).
Beim Schreiben von Programmen ist es bedeutungslos, unter welchem Prozessor die Abarbeitung erfolgen wird. Zwischen Quell- und Objektprogrammen für den Schaltkreis und den Emulator bestehen keine Unterschiede. Die Entscheidung wird erst beim Linken mit der Wahl der Interfacebibliothek getroffen. Demzufolge sind auch die Programme der numerischen Bibliotheken auf beiden Prozessoren abarbeitungsfähig.
Der Leistungsumfang des WM87 entspricht dem in Anlage 6 aufgeführten Befehlssatz. Er beinhaltet Befehle für Transportoperationen, für arithmetische- und Vergleichsoperationen, für transzendente Funktionen sowie Befehle zur Steuerung des Prozessors. Die Arbeit erfolgt auf der Basis von acht 80-Bit-Registern (Register-Stack) in den Datenformaten integer, BCD und Gleitkomma (real).

2.2. Zahlendarstellung

2.2.1. Zahlensystem

Auf Grund der Endlichkeit von Registern und Speicher sind Größe und Genauigkeit der darstellbaren Zahlen im Rechner begrenzt. Daraus ergibt sich ein diskreter und endlicher Zahlenvorrat. Das heißt, das Zahlensystem eines Rechners ist nur eine Untermenge und Näherung des realen Zahlensystems.
Der Zahlenbereich des WM87 erstreckt sich etwa von $\pm 2.3*10^{-308}$ bis $\pm 1.7*10^{308}$ mit einer Genauigkeit von 17 Dezimalstellen in der Datenart long-real. Integer-Zahlen können innerhalb des Bereiches $\pm 2^{64}$ exakt dargestellt werden.

2.2.2. Datenformate

2.2.2.1. Übersicht

Der WM87 unterstützt die drei Datenformate:

- integer
- BCD (packed decimal)
- Gleitkomma (real)

Sie werden in die Datenarten word-, short- und long-integer sowie short-, long- und temporary-real unterteilt.
Die Datenarten und deren Wertebereiche sind in Anlage 1 dargestellt.

2.2.2.2. Codierung der Datenarten

Für alle Datenarten gilt, daß das niederwertigste Bit rechts im Byte mit der niederwertigsten Speicheradresse steht. Das Vorzeichenbit ist generell das höchstwertigste Bit des Bytes mit der höchsten Speicheradresse..
Neben den Bitkombinationen, die zur Zahlendarstellung genutzt werden, gibt es insbesondere im Datenformat real noch weitere zur Darstellung spezieller Werte (siehe Abschnitt 2.3.).
In den Anlagen 2 bis 5 sind die Codierungen der Zahlen und speziellen Werte in den verschiedenen Datenarten enthalten.

2.2.2.3. Integer-Zahlen

Im Datenformat integer stehen drei Datenarten zur Verfügung, die sich nur in der Zahl ihrer signifikanten Bits unterscheiden. Die Darstellung negativer Zahlen erfolgt im Zweierkomplement.
Die Werte -2^{-15}, -2^{-31}, 2^{-63} stellen sowohl den größten negativen Wert als auch den speziellen Wert Indefinite der entsprechenden Datenart dar (siehe Abschnitt 3.8.).

word-integer

```
----------------------
|v| signif. Stellen |
----------------------
15  ...             0
```

short-integer

```
-----------------------------------
|v| signifikante Stellen         |
-----------------------------------
31                               0
```

long-integer

```
----------------------------------------------------------------------
|v| signifikante Stellen                                             |
----------------------------------------------------------------------
63  ....                                                             0
```

2.2.2.4. BCD-Zahlen (packed decimal)

Diese Zahlen werden durch die binäre Codierung einzelner Dezimalziffern dargestellt. Das heißt, mit Hilfe eines Bytes (2 Tetraden) können 2 Dezimalziffern verschlüsselt werden. Die Gesamtlänge einer BCD-Zahl erstreckt sich über 5 Speicherworte (18 Dezimalstellen).
Negative Zahlen werden durch Vorzeichen und Betrag dargestellt. Das höchstwertige Bit ist das Vorzeichenbit. Die Bits 72 ... 78 werden nicht genutzt.

```
--------------------------------------------------------------------
|v|       | d17                                      | d1   | d0 |
--------------------------------------------------------------------
79        72                                           4        0
```

2.2.5. Real-Zahlen (Gleitkommazahlen)

Gleitkommazahlen werden durch Mantisse, Exponent und Vorzeichenbit repräsentiert. Die Länge von Mantisse und Exponent ist abhängig von der Datenart. Die Darstellung von negativen Zahlen erfolgt durch Vorzeichen und Betrag. Die Verarbeitung erfolgt auf der Basis normalisierter Zahlen, das heißt, der gedachte Binärpunkt steht vor dem höchstwertigsten Bit der Mantisse und das Bit vor dem Dezimalpunkt wäre 1. Im Datenformat real stehen drei Datenarten zur Verfügung:

short-real

```
---------------------------------
|v| biased     |               |
| | Exponent   | Mantisse      |
---------------------------------
31            23              0
```

long-real

```
---------------------------------------------------------
|v| biased          |                                   |
| | Exponent        |        Mantisse                   |
---------------------------------------------------------
63                 52                                   0
```

temporary-real

```
-------------------------------------------------------------------------
|v| biased                  |                                           |
| | Exponent                |          Mantisse                         |
-------------------------------------------------------------------------
79                         64                                           0
```

Die Datenart temporary-real kann nur zur internen Zahlendarstellung (Zwischenergebnisse usw.) benutzt werden. Die Anzahl der signifikanten Bits entspricht der WM87-Stack-Registerlänge. Im Gegensatz zu den Formaten short-real und long-real wird hier das Integerbit der Mantisse mit abgebildet. Der gedachte Binärpunkt steht demzufolge nach dem ersten Mantissenbit.

Beim Real-Format wird der Exponent aus dem tatsächlichen Binärexponenten und einem vom speziellen Typ abhängigen Offset gebildet (biased-exponent).
Die Datenarten short-real und long-real existieren nur im Speicher. Wird eine solche Zahl in ein Stack-Register geladen, so erfolgt eine Konvertierung in die Datenart temporary-real, die die Grundlage für alle internen Operationen bildet.
Umgekehrt findet bei Speicheroperationen eine Konvertierung in die Datenarten short- bzw. long-real statt.

2.3. Spezielle Werte

2.3.1. Vorbemerkungen

Neben der zur Zahlendarstellung genutzten Bitkombinationen existieren insbesondere im Datenformat real noch weitere Werte zur Kenntlichmachung bestimmter Zustände wie Infinity, Indefinite, NaN usw. (siehe Anlage 2 bis 3).

2.3.2. Nonnormals (nichtnormalisierte Zahlen)

Die Gleitkommaoperationen erfolgen auf der Basis normalisierter Operanden, bei denen das höchstsignifikante Mantissenbit (Integerbit) gesetzt ist. Bei nicht gesetztem Bit handelt es sich um nichtnormalisierte Zahlen.

2.3.3. Denormals (denormalisierte Zahlen)

Denormals sind das Ergebnis von Operationen, die zu einem Underflow (Exponentenunterschreitung) führten. Er tritt dann ein, wenn eine Zahl zu klein ist, um in der betreffenden Datenart dargestellt werden zu können. Der Prozessor überführt Denormals in Unnormals (unnormalisierte Zahlen), setzt das D-Bit im Statuswort und führt die Operation aus (außer Division). Der Exponent der Denormals ist immer Null.

2.3.4. Unnormals (Unnormalisierte Zahlen)

Unnormals existieren in der Datenart temporary-real. Während der Exponent jeden Wert annehmen kann, ist das Integerbit der Mantisse in jedem Falle rückgesetzt.
Anlage 13 zeigt die Reaktion der Prozessoren bei unnormalisierten Operanden.

2.3.5. Nullen und Pseudonullen

In den Datenformaten real und BCD können Nullen mit negativem Vorzeichen auftreten. Bei der Verarbeitung dieser Werte ist das Vorzeichen bedeutungslos. Real-Werte, deren Mantisse gleich Null und deren Exponent ungleich Null aber kleiner als der maximale Wert ist, werden als Pseudonullen bezeichnet.
Anlage 12 zeigt das Verhalten der Prozessoren bei Null-Operanden.

2.3.6. Infinities

Die Bitkombination mit maximalem Exponenten (alle Bits = 1), gesetztem Integerbit und rückgesetzten übrigen Mantissenbits im Real-Format wird als Infinity (Unendlich) bezeichnet. Das Vorzeichenbit kann 0 oder 1 sein (+ oder - Unendlich).
Mit Hilfe des Infinity-Control-Bits im Steuerwort des Prozessors kann festgelegt werden, ob das Vorzeichen der Infinities berücksichtigt wird (Affin-Modus) oder nicht (Projektiv-Modus).
In Anlage 11 ist die Reaktion des Prozessors auf Infinity-Operanden dargestellt.

2.3.7. NaNs

Werte mit maximalem Exponenten und beliebiger Mantisse im Real-Format werden als NaNs (not a number) bezeichnet. Außer den speziellen Werten Real-Indefinite (siehe Abschnitt 2.3.8.) sind alle NaN vom Anwender zu definieren.
NaN-Operanden erzeugen außer bei transzendenten Befehlen, im Prozessor das Exeception invalid operation (unerlaubte Operation). Ist dieses Exception maskiert, bildet das verursachende NaN das Ergebnis der Operation. Sind beide Operanden NaNs, wird das Resultat durch den betragsmäßig größeren Wert gebildet.
Bei unmaskiertem I-Exception erfolgt ein Aufruf des Exception-Handlers. Die Operanden sind unverändert.
Auf Grund ihrer speziellen Eigenschaften eignen sich NaNs zur Verschlüsselung zusätzlicher Informationen insbesondere zur Fehlerortung.

2.3.8. Indefinities

Indefinities sind datenartspezifisch. Sie werden durch den Prozessor beim Auftreten von maskierten invalid operations als Resultat erzeugt. Im Datenformat real stellen die Indefinities einen speziellen Wert der NaNs dar. Sie werden nur dann als Ergebnis gesetzt, wenn die Ursache des I-Exceptions kein NaN war.

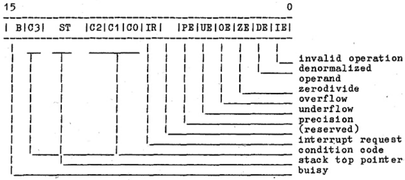
2.4. Befehlssatz der Prozessoren

Die Prozessoren realisieren Befehle in den Datenarten integer, BCD und Gleitkomma. Während für BCD lediglich Transportbefehle zur Verfügung stehen, liegen für integer und Gleitkomma auch Arithmetik- und Vergleichsbefehle, für Gleitkomma zusätzlich Befehle zur Berechnung transzendenter Funktionen vor.
Der Befehlssatz umfaßt weiterhin Befehle zur Steuerung der Prozessoren wie Initialisierung und Exception-Behandlung.
Eine Zusammenstellung aller WM87-Befehle ist in Anlage 6 enthalten. In der "Anleitung für den Bediener und Programmierer, Programmpaket für modulare Programmierung" werden die Befehle erläutert.

2.5. Register der Prozessoren

2.5.1. Statuswort

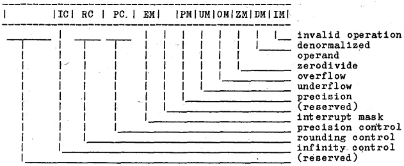
Das Statuswort liefert ein Abbild des Prozessorzustandes. Es kann durch entsprechende Befehle in den Speicher bzw. die Register gebracht und ausgewertet werden.



Die Bitpositionen 0 bis 5 zeigen das Auftreten bestimmter Exceptions bei der Befehlsabarbeitung an. Das Bit 7 ist das Interrupt-Request-Bit. Es wird vom Prozessor gesetzt. Durch verschiedene Befehle (FCOM u.a.) werden die Condition-Code-Bits 8, 9, 10 und 14 beeinflußt.
Die Positionen 11, 12 und 13 enthalten den Stack-Top-Pointer (ST). Er zeigt auf die aktuelle Register-Stack-Position (siehe Abschnitt 2.5.4.). Bit 15 ist gesetzt, wenn der Prozessor aktiv ist.

2.5.2. Steuerwort

Das Steuerwort dient zur Beeinflussung der Befehlsabarbeitung. Es besteht aus der Exception-Maske, der Interrupt-Maske und weiteren Steuerbits.



Die Bitpositionen 0 bis 5 beziehen sich auf die im Statuswort angezeigten Exceptions. Die Steuerung erfolgt über Setzen bzw. Rücksetzen der betreffenden Bits im Steuerwort (Maskenbits). Ist beim Auftreten eines Exceptions das entsprechende Maskenbit rückgesetzt (Exception unmaskiert), so erfolgt über den Interrupt-Vektor 16 (40H) ein Sprung in den zugehörigen Exception-Handler. Bei maskiertem Exception wird ein Standardwert erzeugt und die Operation fortgesetzt. In Anlage 10 sind alle möglichen Exceptions, deren Ursachen und die Reaktion des Prozessors aufgeführt.

Die Bitposition 7 ist die Interrupt-Enable-Maske. Wenn Bit 7 gesetzt ist (maskiert), dann werden Interrupts unterdrückt.
Über die Bitpositionen 8 und 9 kann die Genauigkeit der Berechnungen beeinflußt werden. Es gilt folgende Zuordnung:

00 = 24 Bit
01 = reserviert
10 = 53 Bit
11 = 64 Bit

Wenn keine speziellen Forderungen vorliegen, sollte mit maximaler Genauigkeit gearbeitet werden, weil dadurch die Mantissenlänge voll genutzt wird und keine Nachteile bzgl. der Operationszeit auftreten.
Die Bitpositionen 10 und 11 dienen zur Steuerung der bei verschiedenen Operationen erforderlichen Rundung. Folgende Möglichkeiten stehen zur Verfügung:

00 – Runden auf den nächstliegenden Wert.
Bei gleichen Abständen wird das Resultat gleich dem Wert gesetzt, dessen niederwertigstes Bit Null ist.

01 – Das Resultat wird auf den kleineren Wert abgerundet.

10 – Das Resultat wird auf den größeren Wert aufgerundet.

11 – Das Resultat wird gleich der nächst kleineren Integer-Zahl gesetzt.

Bit 12 dient der Behandlung von Infinities. Während im Projectiv-Modus (Bit 12 = 0) das Vorzeichen unberücksichtigt bleibt, wird im Affin-Modus zwischen +Infinity und -Infinity unterschieden. Gegenüber den Ergebnissen im Projectiv-Modus liefern die im Affin-Modus mehr, zum Teil aber auch Fehlinformationen.

2.5.3. Register-Stack

Der WM87-Prozessor besitzt 8 Register (ST0 ... ST7) zu je 80 Bit Länge. Sie befinden sich im Stack. Die aktuelle Stackposition (stack-top) wird im ST-Feld des Statuswortes angezeigt (siehe Abschnitt 2.5.1.).
Ladeoperationen vermindern die aktuelle Stackposition um 1 und bringen den Wert auf die neue Position. Speicheroperationen greifen zur aktuellen Position zu und erhöhen anschließend um 1. Es ist zu beachten, daß ein Ladebefehl bei ST = 000B zu ST = 111B und umgekehrt ein Speicherbefehl bei ST = 111B zu ST = 000B führen. Entsprechend dem WM86-Stack steigen die Stackpositionen mit den Speicheradressen.
Die Register können implizit oder explizit adressiert werden, d. h., daß bei bestimmten Operationen die Registerangabe entfallen kann. Es wird dann auf die aktuelle Position zugegriffen (FSQRT u.a.). Andere Operationen verlangen explizit die Angabe von Registern. Hierbei ist zu beachten, daß die Adressierung immer relativ zur aktuellen Registerposition laut ST im Statuswort erfolgt. Ist z. B. ST = 3, dann addiert

```
FADD ST,ST2
```

die Inhalte der Register 3 und 5. Die Summe steht im Register 3.

2.6. Exceptions der Prozessoren

2.6.1. Exceptions und deren Ursachen

Während der Befehlsabarbeitung findet eine Prüfung auf die folgenden Exceptions statt.

- invalid operation
- overflow
- underflow
- zerodivide
- denormalized operand
- precision

Bei Auftreten eines Exceptions wird das entsprechende Bit im Statuswort des WM87 gesetzt (siehe Abschnitt 2.5.1.). Das Exception invalid operation kann verursacht werden durch:

- Stacküberlauf beim Laden eines Registers
- Stackunterlauf beim Speichern eines Registers
- Argument ist ein NaN
- indeterminierte Ergebnisse (0/0, Wurzel aus einer negativen Zahl usw.)

Es wird in jedem Fall durch einen Programmfehler verursacht.
Die Ergebnisse overflow und underflow weisen auf den Exponenten hin. Das Resultat ist zu groß bzw. zu klein, um in der entsprechenden Datenart dargestellt werden zu können.
Beim Versuch, eine Zahl ungleich Null durch Null zu dividieren, wird das Exception zerodivide angezeigt.
Denormalisierte Argumente (Denormals) führen zum Setzen von Bit 2 im Statuswort.
Wenn ein Ergebnis in der geforderten Datenart nicht genau dargestellt werden kann, wird die Zahl gerundet. In Verbindung damit wird das Bit 5 im Statuswort zur Anzeige eines Genauigkeitsverlustes genutzt.
Während die Exceptions invalid operation, zerodivide und denormalized operand vor Beginn der Operation angezeigt werden, stehen die restlichen Exceptions erst mit dem Resultat zur Verfügung. Weiterhin ist der Zeitpunkt des Auftretens des Exceptions und damit des Abbruchs der Operation entscheidend dafür, ob Register-Stack und Speicher durch den Befehl bereits geändert wurden. Beim Auftreten von Exceptions, die vor der eigentlichen Befehlsabarbeitung angezeigt werden, sind Register-Stack und Speicher unverändert.

Beim Auftreten von Mehrfach-Exceptions gelangt nur das gemäß der nachfolgenden Prioritätsliste höchstwertigste zur Anzeige.

- denormalized operand (wenn unmaskiert)
- invalid operation
- zerodivide
- denormalized operand (wenn maskiert)
- overflow/underflow
- precision

Das bedeutet, daß z. B. die Division Null durch Null als invalid operation und nicht als zerodivide angezeigt wird.

2.6.2. Auswertung der Exceptions

Es gibt grundsätzlich zwei Möglichkeiten zur Reaktion auf im Statuswort angezeigte Exceptions.
Die erste Möglichkeit besteht in der Nutzung des Trap-Mechanismus. Dabei kann beim Auftreten von Exceptions über den Trap-Vektor 16 ein Exception-Handler erreicht werden. Mit dem Steuerwort können diejenigen Exceptions festgelegt werden, die einen Trap verursachen sollen (unmaskierte Exceptions).
Die zweite Möglichkeit verzichtet auf die Trap-Nutzung. Beim Auftreten von Exceptions, deren Bit im Steuerwort gesetzt ist (maskierte Exceptions), erzeugt der Prozessor Standardwerte und setzt im Programm fort. Anlage 8 enthält die Reaktion des Prozessors auf maskierte Exceptions.
Es erweist sich als günstig, sowohl mit maskierten als auch mit unmaskierten Exceptions zu arbeiten.

2.6.3. Exception-Handler

In Anlage 10 sind die möglichen Exceptions, deren Ursachen und die Reaktionen des Prozessors bei gesetztem und ungesetztem Maskenbit enthalten. Der Exception-Handler muß differenziert mit maskierten und unmaskierten Exceptions arbeiten. Es ist sinnvoll, alle Exceptions außer invalid operation, das in jedem Falle auf einen Programmfehler hinweist, zu maskieren. Der Prozessor erzeugt in all diesen Fällen Standardwerte, die eine sinnvolle Weiterarbeit ermöglichen.
Der Exception-Handler sollte aus den Teilen Vorbereitung, Ausführung und Nachbereitung bestehen. Im Teil Vorbereitung müssen die Maßnahmen durchgeführt werden, die gesperrte Interrupts voraussetzen. Dazu gehören das Retten von Registern und des WM87-Status. Danach können Interrupts erlaubt werden.
Der Teil Ausführung muß eine Auswertung der Registerinhalte und Statusinformationen beinhalten und zu einer anwendungspezifischen Beantwortung der Exceptions führen. Das kann sowohl einen Abbruch des Programmes, eine Fehlermeldung oder die Fortsetzung des Programms bedeuten.
Der Programmteil Nachbereitung muß die geretteten Registerinhalte und Statusinformationen aktualisieren, die Exception-Bits im Statuswort löschen und die Rückkehr ins Hauptprogramm realisieren.
Die Exception-Handler-Bibliothek EH87 enthält Routinen, die beim Schreiben von Exception-Handlern genutzt werden können.

2.7. Arbeit mit den Prozessoren

2.7.1. Arbeit mit dem Emulator EWM87.OBJ

Im Emulator sind sowohl die Befehle als auch die Register des Schaltkreises WM87 im WM86-Speicher nachgebildet. Die Berechnungen erfolgen auf der Basis der Algorithmen des Schaltkreises.
Das hat die Übereinstimmung der Ergebnisse zur Folge. Der Emulator belegt etwa 16K-Byte im Code-Segment, 150 Byte im Data-Segment und 60 Byte im Stack-Segment. Folgende Hinweise sind beim Schreiben eines Programmes, das WM87-Befehle enthält und unter den Emulator arbeiten soll, zu beachten:

- Die Interrupts 20 bis 31 (Adressen 50H bis 7FH) werden vom Emulator belegt (durch INIT87 gesetzt).

- An den Interrupt 16 (Adressen 40H bis 43H) ist ein Exception-Handler anzuschließen (wenn unmaskierte Exceptions auftreten).
- Erlauben von Interrupts für unmaskierte Exceptions (STI).
- FWAIT wird umgewandelt in NOP. WAIT bleibt erhalten und führt zu endlosem Warten.
- Das Stack-Segment des Anwenderprogramms muß als Segment- und Klassifikationsnamen STACK tragen.
- Zu Beginn des Anwenderprogrammes, spätestens jedoch vor der ersten Prozessornutzung, muß das Initialisierungsprogramm INIT87 aufgerufen werden. Der Status des Prozessors nach der Initialisierung ist in der Anlage 7 dargestellt.

Innerhalb des LINK86-Kommandos ist diese Reihenfolge der Objektmoduln und Bibliotheken einzuhalten:

Anwenderprogramm
DCON87.L86 (falls genutzt)
CEL87.L86 (falls genutzt)
EH87.L86 (falls genutzt)
EWM87.OBJ
EWM87.L86

Als Interfacebibliothek ist EWM87.L86 zu nutzen.

2.7.2. Arbeit mit dem Schaltkreis WM87

Bei Nutzung des Schaltkreises WM87 sind im Anwenderprogramm folgende Maßnahmen zu treffen:

- Aufruf des Initialisierungsprogramms INIT87 vor Abarbeitung des ersten WM87-Befehls.
- Anschluß eines Exception-Handlers an Interrupt 16, wenn unmaskierte Exceptions auftreten können.
- Erlauben von Interrupts für unmaskierte Exceptions (STI).

Im LINK86-Kommando ist diese Reihenfolge der Programmoduln einzuhalten:

Anwenderprogramm
DCON87.L86 (falls genutzt)
CEL87.L86 (falls genutzt)
EH87.L86 (falls genutzt)
WM87.L86

Als Interfacebibliothek ist WM87.L86 zu nutzen.

3. Interfacebibliotheken

Numerische Operationen können auf der Basis des Schaltkreises WM87 oder dessen programmtechnischen Simulators (Emulator WM87) erfolgen. Die Entscheidung über den gewählten Prozessor wird beim Linken mit der Festlegung der Interfacebibliothek getroffen. Die Interfacebibliotheken stellen die Verbindung zwischen dem Prozessor und den ihn aufrufenden Programmen her.

Funktion

Alle Übersetzer, die den Befehlssatz des WM87 unterstützen, liefern bei WM87-Befehlen den Maschinenbefehl in Verbindung mit einer globalen Referenz, deren Werte in den Interfacebibliotheken definiert sind. In Abhängigkeit davon, welche Anschlußbibliothek beim Linken angegeben wurde, bleibt der vom Übersetzungsprogramm erzeugte Maschinenbefehl erhalten (Schaltkreisvariante) oder er wird in einen Interrupt überführt (Emulatorvariante). Insgesamt werden 12 Interruptvektoren (20 bis 31) belegt. Die zugehörigen Emulatorroutinen stellen die Operanden bereit und führen den Maschinenbefehlen entsprechende Operationen durch.
Die Interfacebibliotheken enthalten die Programme INIT87. Sie dienen zur Initialisierung der Prozessoren.

Interfacebibliothek WM87.L86

Diese Bibliothek ist bei der Arbeit mit dem Schaltkreis WM87 im LINK86-Kommando anzugeben. Sie bewirkt die Auflösung der globalen Referenzen. Durch Aufruf von INIT87 im Anwenderprogramm erfolgt die Initialisierung des Schaltkreises und das Setzen des Steuerwortes.

Interfacebibliothek EWM87.L86

Diese Bibliothek ist bei der Arbeit mit dem Emulator EWM87.OBJ im LINK86-Kommando anzugeben. Sie bewirkt die Auflösung der globalen Referenzen. Das Programm INIT87 setzt die Interruptvektoren 20 bis 31 und initialisiert den Emulator.

Interfacebibliothek 87NULL.L86

Diese Bibliothek muß genutzt werden, wenn das Anwenderprogramm keine WM87-Befehle sondern lediglich Aufrufe der Initialisierungsroutine INIT87 enthält.

Nutzung der Interfacebibliotheken

Die Interfacebibliotheken müssen im LINK86-Kommando nach allen Programmen stehen, die WM87-Befehle enthalten.

4. Konvertierungsbibliothek DCON87

4.1. Vorbemerkungen

Rechnerintern stehen drei verschiedene Datenarten für das Gleitkommaformat zur Verfügung. Die Routinen der Bibliothek DCON87 realisieren die Konvertierung dieser internen Datenarten in die externe Darstellung und umgekehrt. Weiterhin sind Programme zur Konvertierung der Datenarten untereinander vorhanden.

Die Namen der Programme beginnen mit mqc, um weitgehend auszuschließen, daß im Anwenderprogramm gleiche Namen benutzt werden. Durch die Kleinschreibung wird eine bessere Lesbarkeit erreicht.

Das Programm mqeBIN_DECLOW konvertiert eine Gleitkommazahl beliebiger Datenart von der rechnerinternen in die externe Darstellung. Dabei ist zu berücksichtigen, daß keine endgültige Fassung der Zahl erzeugt wird. Stattdessen werden Zeichenblöcke mit Dezimalziffern, Exponenten und Vorzeichen bereitgestellt, aus denen der Anwender die von ihm gewünschte Form für die Ausgabe herstellen kann.

Das Programm mqcDEC_BIN konvertiert eine Gleitkommzahl von der externen in die interne Darstellung. Die externe Zahl kann aus einer Dezimalzahl mit oder ohne Vorzeichen und Punkt sowie einem Exponenten bestehen.

Das Programm mqcDECLOW_BIN konvertiert Dezimalzahlen, die der durch mqcBIN_DECLOW erzeugten Form entsprechen, in eine rechnerinterne Form.
Die Programme mqcLONG_TEMP, mqrSHORT_TEMP, mqc_TEMP_LONG und mqcTEMP_SHORT realisieren Konvertierungen innerhalb der internen Datenarten. Dabei werden NaNs erzeugt, wenn auch die Ausgangswerte NaNs sind.

In Anlage 15 sind die globalen Symbole aufgeführt, die von einigen Übersetzern genutzt werden. Der Anwender sollte diese Namen nicht benutzen.

4.2. Vereinbarungen im Assemblerprogramm

In jedem Quellprogramm, das Rufe zu DCON87-Programmen enthält, sind die Namen der Programme als externe Symbole vom Typ FAR zu vereinbaren. Es sollten nur die tatsächlich genutzten Programme vereinbart werden.
Weiterhin sind die Werte der Datenarten und das Parameterfeld zu definieren.
Im folgenden sind die zu treffenden Vereinbarungen aufgeführt.

```
; RASM86-Definitionen für die Nutzung der
; Konvertierungsbibliothek DCON87
; Die folgenden EXTERN-Vereinbarungen müssen außerhalb aller
; Segmentvereinbarungen stehen:
                EXTERN   mqcBIN_DECLOW:FAR
                EXTERN   mqcDEC_BIN:FAR
                EXTERN   mqcDECLOW_BIN:FAR
                EXTERN   mqcLONG_TEMP:FAR
                EXTERN   mqcSHORT_TEMP:FAR
                EXTERN   mqcTEMP_LONG:FAR
                EXTERN   mqcTEMP_SHORT:FAR
SHORT_REAL      EQU   0
LONG_REAL       EQU   2
TEMP_REAL       EQU   3
;
; Die folgenden Vereinbarungen sind im Data-Segment zu treffen.
                DSEG
BIN_PTR         DW    0,0
BIN_TYPE        DB    0
DEC_LENGTH      DB    0
DEC_PTR         DW    0,0
DEC_EXPONENT    DW    0
DEC_SIGN        DB    0
;
```

4.3. Stacknutzung

Die Programme der DCON87 benötigen 176 Byte im Stack. Dieser Bereich wird durch die Bibliothek selbst definiert.
Bei Eintritt in die DCON87-Routinen wird der gesamte Status des WM87-Prozessors gerettet und bei Verlassen wieder aktualisiert.

4.4. Genauigkeit der Konvertierungsalgorithmen

DCON87 garantiert, daß eine 18-stellige Dezimalzahl durch Konvertieren in die Datenart temporary-real und anschließendes Rückkonvertieren unverändert bleibt.

Es ist jedoch nicht gewährleistet, daß eine Zahl der Datenart temporary-real durch Konvertieren zu einer Dezimalzahl und nachfolgender Rückkonvertierung nicht verändert wird. Das würde eine Genauigkeit erfordern, die über die der Datenart temporary-real hinausgeht. Eine Arithmetik dieser Genauigkeit hätte aber wiederum verminderte Operationsgeschwindigkeiten zur Folge.
Der Genauigkeitsverlust beträgt maximal 3 Bit.
Durch die Programme der DCON87 werden in Abhängigkeit vom Dezimalexponenten folgende Genauigkeiten erreicht:

maximal mögliche Genauigkeit:

Konvertierungsart	signifikante Stellen	Exponent
dezimal-->short-real	9	13
dezimal-->long-real	19	27
short-real-->dezimal	9	13
long real-->dezimal	17	27

Das bedeutet z. B., daß der Zahlenbereich für eine exakte Konvertierung in der Datenart short-real zwischen +1E-13 und 999999999E13 liegt.

maximaler Genauigkeitsverlust 0,47 mal letzte Ziffer:

Konvertierungsart	signifikante Stellen	Exponent
dezimal-->short-real	9	99
dezimal-->long-real	19	999
short-real-->dezimal	9	54
long-real-->dezimal	17	341

4.5. Exceptions der DCON87

Die Programme der DCON87 nutzen den Fehlermeldungsmechanismus des WM87. Sobald ein Fehler auftritt, wird das entsprechende Bit im Statuswort des WM87 gesetzt. Unter Nutzung der WM87-Befehle kann durch maskierte und unmaskierte Exceptions im Statuswort sowie durch einen entsprechenden Exception-Handler auf die verschiedenen Exceptions differenziert reagiert werden. Auf diese Weise erübrigt sich ein Test auf Fehlerbedingungen nach jedem Aufruf eines DCON87-Programms.

Bei jedem unmaskierten Exception, das durch ein DCON87-Programm angezeigt wird, werden vor dem Aufruf des Exception-Handlers die folgenden Informationen in den WM87-Registern abgelegt:

- Der Operationscode OOL OLLO LLLO (16EH) wird in das 11-Bit-Operationscoderegister des WM87 gebracht. Diesem Operationscode entspricht der Befehl FLDCW, der selbst das Operationscoderegister nicht ändert. Diese Information weist zusammen mit einer Befehlsadresse, in der nicht alle Bit gesetzt sind, darauf hin, daß das betreffende Exception innerhalb einer DCON87-Routine festgestellt wurde. (Eine Befehlsadresse mit gesetzten Bits deutet auf ein Programm der CEL87 hin.)

- Die 20-Bit-Adresse der ereignisverursachenden DCON87-Routine befindet sich im Befehlszähler des WM87.

- Die 20-Bit-Adresse der zu konvertierenden Zahl steht im Operanden-Adreßregister des WM87. Für die Programme mqcDEC_BIN und mqcDECLOW_BIN ist es die Adresse eines internen DCON87-Puffers im WM86-Stack, der die Dezimalziffern in der für mqcDECLOW_BIN erforderlichen Form enthält. Für alle anderen Programme ist es die wahre Adresse der zu konvertierenden Binärzahl.

Eine weitere Möglichkeit zur Fehleranzeige ist die Auswertung des AL-Registers. Es enthält bei Rückkehr ins aufrufende Programm das Exception-Byte des Statuswortes.

4.6. Format der Dezimalzahlen bei mqcDEC_BIN

Die Ziffern und Zeichen der mittels mqcDEC_BIN zu konvertierenden Zahl müssen in fortlaufender Folge im KOI7-Code im Speicher bereitstehen. Die Adresse des ersten Bytes muß bei Aufruf des Programms im Parameterblock enthalten sein.

Leerzeichen sind am Ende der Dezimalzahl nicht aber zu Beginn oder innerhalb der Zahl erlaubt. Die Zahl kann mit "+" oder "-" beginnen. Zahlen ohne Vorzeichen werden als positiv definiert.

Nach dem Vorzeichen stehen die einzelnen Ziffern mit oder ohne Dezimalpunkt. Der Dezimalpunkt kann innerhalb, vor oder nach den Ziffern erscheinen.
Es muß mindestens eine Ziffer vorhanden sein. Die maximale Ziffernanzahl beträgt 255. Es werden jedoch nur die ersten 20 (außer führende Nullen) berücksichtigt.

Den Ziffern folgt wahlweise ein Exponent. Er besteht aus einem Buchstaben (E, D oder T), einem optionalen Vorzeichen und einer Anzahl von Ziffern. Der Exponent gibt an, mit welcher Zehnerpotenz die davorstehende Zahl zu multiplizieren ist. Die Exponentenkennzeichen sind gleichwertig. Es können sowohl Groß- als auch Kleinbuchstaben sein.

Beispiele für Dezimalzahlen

gültige Zeichenfolgen:

```
100
+100
100.
100.00
2000.000000000000000000012345
1E2
1T3
1d4
1.E5
.1e6
3.141159265
-16
-34.12345
5.6789
.001
-0.6
-0
```

fehlerhafte Zeichenfolgen:

```
1  E2   ; Leerzeichen sind unzulässig
E1      ; Mantisse fehlt
.T6     ; Mantisse fehlt
```

4.7. Linken der Programme

Die Programme zur Konvertierung der Gleitkommazahlen sind in der Bibliothek DCON87 enthalten. Infolgedessen ist der Name dieser Bibliothek im LINK86-Kommando anzugeben. DCON87 nutzt entweder den Schaltkreis WM87 oder dessen Emulator EWM87. Entsprechend sind die zugehörigen Interfacebibliotheken WM87.L86 bzw. EWM87.L86 notwendig.
Es ist folgende Reihung der Objektmodule im LINK86-Kommando einzuhalten:

```
Anwenderobjektprogramm
DCON87.L86
CEL87.L86 (wenn erforderlich)
EH87.L86 (wenn genutzt)
WM87.L86 (wenn Schaltkreis genutzt)
EWM87, EWM87.L86 (wenn Emulator genutzt)
```

Beispiel

Die Programme AN1 und AN2 seien Anwenderobjekte.

```
A>LINK86  C:AN.CMD = C:AN1, C:AN2, DCON87.L86
CEL87.L86, EH87.L86, EWM87, EWM87.L86 (CR)
```

4.8. Beschreibung der Konvertierungsprogramme

4.8.1. Konvertierung binär in dezimal mqcBIN_DECLOW

Parameter

BIN_DECLOW_BLOCK_PTR enthält die 2-Wort-Adresse des Parameterblockes, der aus den folgenden 6 Einzelparametern besteht:

BIN_PTR ist ein Doppelwort, das die Adresse der Binärzahl enthält. Die Länge des Feldes der Binärzahl wird durch deren Typ (BIN_TYP) festgelegt.

BIN_TYPE ist ein Bytewert, der die Codierung der Datenart enthält. Als Datenarten sind short-real (0), long-real (2) oder temporary-real (3) zulässig. Andere Werte als 0, 2 oder 3 führen zu undefinierten Ergebnissen.

DEC_LENGTH ist ein Bytewert, der die Anzahl der auf dem Bereich DEC_PTR zu erzeugenden Dezimalziffern enthält. Alle Bytewerte sind erlaubt. Bei DEC_LENGTH = 0 werden nur DEC_EXPONENT und SIGN erzeugt. DEC_PTR bleibt unverändert. Wird DEC_LENGTH = 0 bei Ausnahmewerten angegeben, erfolgt eine Fehlermeldung. Ausnahmewerte sind NaN, Indefinite und ±0.

DEC_PTR ist ein Doppelwort, ds die Adresse des Bereiches der Dezimalziffern enthält. Exponent und Vorzeichen sind darin nicht enthalten.

DEC_EXPONENT ist ein Wortparameter. Er enthält den Exponenten der Zehnerpotenz, mit der die in DEC_PTR enthaltenen Ziffern zu multiplizieren sind, um den wahren Wert

zu erhalten. Es wird vorausgesetzt, daß der Dezimalpunkt nach der letzten Dezimalziffer in DEC_PTR steht. Für NaN und Infinities ist der Dezimalexponent 32767. Für +0 und -0 ist er 0.

DEC_SIGN — ist ein Bytewert. Er enthält das Vorzeichen der Dezimalzahl im ASCII-Code. Bei positiven Zahlen ist er 2BH, bei negativen 2DH.

Bei Ausnahmewerten erzeugte Zeichenfolgen:

Ausnahmewert	DEC_SIGN	Dezimalziffernbereich
NaN	"."	"." gefolgt von Leerzeichen
+Infinity	"+"	"+" gefolgt von Leerzeichen
-Infinity	"-"	"-" gefolgt von Leerzeichen
+0	"0"	Leerzeichen
-0	"-"	"0" gefolgt von Leerzeichen

Die 2-Wort-Adresse des Parameterblockes ist vor dem Aufruf von mqc_BIN_DECLOW auf den Stack zu bringen.

Funktion

mqcBIN_DECLOW konvertiert die im Bereich BIN_PTR stehende Gleitkommazahl vom Typ BIN_TYPE in eine Dezimalzahl, die, getrennt in Dezimalziffern, Exponent und Vorzeichen, auf den Bereichen DEC_PTR, DEC_EXPONENT und DEC_SIGN zur Verfügung steht. Der Anwender kann aus diesen Folgen das gewünschte Format erzeugen. mqcBIN_DECLOW liefert 18 Dezimalstellen. Wenn DEC_LENGTH größer als 18 ist, werden 18 Dezimalziffern, gefolgt von einer entsprechenden Anzahl Underscores (Unterstrich), erzeugt. Der Dezimalpunkt wird als hinter dem letzten Underscores stehend angenommen. Auf diese Weise können Underscores als unbekannte Dezimalziffern interpretiert werden.

Fehler

Es können drei unterschiedliche Fehler auftreten:

unerlaubter Eingangswert:
Dieser Fehler erscheint, wenn die spezifizierte DEC_LENGTH = 0 und der binäre Wert minus 0, ein Infinity oder ein NaN ist. In diesen Fällen enthält der Bereich für Dezimalziffern keine Ziffern. DEC_SIGN und DEC_EXPONENT sind entsprechend gesetzt. Bei unmaskierten I-Exception steht der binäre Wert bei Aufruf des Exception-Handlers auf dem WM87-Stack.

denormalisierter Eingangswert:
Dieser Fehler tritt nur dann auf, wenn BIN_TYPE temporary-real und die führenden Mantissenbits Null sind. DEC_EXPONENT und DEC_SIGN enthalten die richtigen Ergebnisse. Die Folge der Dezimalziffern wird zur Anzeige des denormalisierten Eingangswertes mit "0" eingeleitet. Bei unmaskierten D-Exception steht bei Erreichen des Exception-Handlers der Eingangswert auf dem Register-Stack.

ungenaues Ergebnis:
Der gerundete Binärwert wird konvertiert. Das Format der Dezimalzahl ist korrekt. Bei unmaskiertem P-Exception steht

bei Aufruf des Exception-Handler der Binärwert auf dem Register-Stack -

Programmbeispiel

```
; Die EXTERN-Vereinbarung muß außerhalb der Segmentvereinbarung
; stehen.

          EXTERN                   mqcBIN_DECLOW:PAR
;

          DSEG
BIN_DECLOW_BLOCK_PTR  DD  BIN_PTR   ; Adr. des Parameterpuffers

BIN_PTR        DW   0,0             ; Adr. der Binärzahl
BIN_TYPE       DB   0               ; Datenart
DEC_LENGTH     DB   0               ; Anzahl der Dezimalziffern
DEC_PTR        DW   0,0             ; Adresse der Dezimalziffern
DEC_EXPONENT   DW   0               ; Dezimalexponent
DEC_SIGN       DB   0               ; Vorzeichen
;
               .
               :
;
               CSEG
               .
               :
               MOV   DX,SEG BIN_PTR
               PUSH  DX
               MOV   DX,OFFSET BIN_PTR
               PUSH  DX
               CALLF mqcBIN_DECLOW
```

4.8.2. Konvertierung dezimal in binär mqcDEC_BIN

Parameter

DEC_BIN_BLOCK_PTR enthält die 2-Wort-Adresse des Parameterblockes, der aus folgenden vier Einzelparametern besteht:

BIN_PTR ist ein Doppelwort, das die Adresse der Binärzahl enthält. Die Größe des Feldes der Binärzahl ist abhängig von der Datenart (BIN_TYPE).

BIN TYPE ist ein Bytewert, der die Datenart enthält. Als Datenarten sind short-real (0), long-real (2) und temporary-real (3) zulässig. Andere Werte als 0, 2 oder 3 führen zu undefinierten Ergebnissen.

DEC_LENGTH ist ein Bytewert, der die Länge der im Bereich DEC_PTR enthaltenen Dezimalpunkte enthält. DEC_LENGTH darf alle Werte zwischen 1 und 255 (einschließlich) annehmen. Null führt zu einem undefinierten Ergebnis.

DEC_PTR ist ein Doppelwort, das die Anfangsadresse des Bereiches der Dezimalzahl enthält. Die möglichen Formate der Dezimalzahlen sind unter Abschnitt 4.8.1. erläutert. Der Anwender sollte vor dem Aufruf von

mqcDEC_BIN eine Prüfung der Syntax vornehemen.
Die 2-Wort-Adresse des Parameterblockes ist vor dem Aufruf von mqcDEC_BIN auf den Stack zu bringen.

Funktion

mqcDEC_BIN konvertiert die auf dem Bereich DEC_PTR stehende Dezimalzahl in eine Gleitkommazahl vom Typ BIN_TYPE, die im Bereich BIN_PTR zur Verfügung steht.

Fehler

Die folgenden sechs Fehler sind möglich:

unerlaubter Eingangswert:
Bei syntaktisch fehlerhafter Dezimalzahl wird ein undefinierter Binärwert erzeugt. Es erfolgt keine Fehleranzeige.

Overflow:
In der internen Datenart temporary-real trat ein Überlauf auf. Bei maskiertem O-Exception wird ein vorzeichenbehaftetes Infinity als Gleitkommazahl erzeugt. Bei unmaskiertem O-Exception wird der Exception-Handler aufgerufen. Der aktuelle Register-Stack anthält Indefinity. Der Ausgabepuffer wurde nicht verändert.

Overflow:
In der auszugebenden Datenart, aber nicht in temporary-real, trat ein Überlauf auf. Wenn das O-Exception maskiert ist, wird im Ausgabepuffer ein vorzeichenbehaftetes Infinity erzeugt. Bei unmaskierten Exception wird der Exception-Handler aufgerufen. Die aktuelle Register-Stack-Position enthält den auf die Genauigkeit der auszugebenden Datenart gerundeten Wert.

Underflow:
Während der Konvertierung wurde der Wertebereich der Datenart temporary-real unterschritten. Bei rückgesetztem Maskenbit im Steuerwort des Prozessors wird ein Indefinity auf den Register-Stack gebracht, der Ausgabebereich nicht verändert und der Exception-Handler aufgerufen. Bei maskiertem U-Exception enthält der Ausgabepuffer das Ergebnis des Underflows.

Underflow:
Der Wertebereich der auszugebenden Datenart wurde unterschritten. Auf dem Ausgabepuffer steht bei maskierter Ausnahmebedingung das Ergebnis des Underflows. Bei rückgesetztem Maskenbit wurden der entsprechend der Datenart gerundete Wert auf den Register-Stack gebracht und der Exception-Handler aufgerufen.

Genauigkeitsverlust:
Der Ausgabepuffer enthält das gerundete Ergebnis der Konvertierung. Zur Anzeige der erfolgten Rundung wurde das P-Bit im Statuswort gesetzt. Da der Übergang vom bei der Konvertierung benutzten Format temporary-real in short- bzw. long-real prinzipiell mit einer Vereinigung der signifikanten Stellen verbunden ist, sollte das Auftreten des P-Exceptions nicht als Fehler gewertet werden und das entsprechende Bit im Steuerwort

gesetzt sein.
Bei rückgesetztem Steuerbit enthält der Register-Stack des Exception-Handlers den ursprünglichen Wert.

Programmbeispiel

```
; Die EXTERN-Vereinbarung muß außerhalb der Segment-
; vereinbarungen stehen.
                        EXTERN mqcDEC_BIN: FAR
;
                        DSEG

DEC_BIN_BLOCK_PTR DD    BIN_PTR        ; Adresse des Parameter-
                                       ; puffers
BIN_PTR           DW    0,0            ; Adresse der Binärzahl
BIN_TYPE          DB    0              ; Datenart
DEC_LENGTH        DB    0              ; Anz. der Dezimalzeichen
DEC_PTR           DW    0,0            ; Adr. der Dezimalzahl
                  .
                  :
                  CSEG
                  .
                  :
                  MOV   DX,SEG BIN_PTR
                  PUSH  DX
                  MOV   DX,OFFSET BIN_PTR
                  PUSH  DX
                  CALLF mqcDEC_BIN
                  .
                  :
```

4.8.3. Konvertierung dezimal in binär mqcDECLOW_BIN

Parameter

DECLOW_BIN_BLOCK_PTR enthält die 2-Wort-Adresse des Parameterblockes, der aus folgenden sechs Einzelprametern besteht:

BIN_PTR ist die 2-Wort-Adresse der Binärzahl. Die Anzahl der durch die Gleitkommazahl belegten Bytes ist von der Datenart abhängig.

BIN_TYPE ist ein Bytewert, der die Codierung der Datenart enthält. Als Datenarten sind short-real (0), long-real (2) und temporary-real (3) zulässig. Andere Werte als 0, 2 und 3 führen zu undefinierten Ergebnissen.

DEC_LENGTH ist ein Byteparameter, der die Anzahl der Dezimalziffern enthält. Er darf Werte zwischen 1 und 21 (einschließlich) annehmen. Andere Werte führen zu undefinierten Ergebnissen.

DEC_PTR enthält die 2-Wort-Adresse des Bereiches der Dezimalziffern. Führende Nullen, Dezimalpunkt, Vorzeichen, Exponent und Nachnullen sind darin enthalten.

DEC_EXPONENT enthält den Dezimalexponenten der zu konvertierenden Zahl. Es wird vorausgesetzt, daß der Dezimalpunkt nach der letzten Ziffer der Dezimalzahl steht. Negative Exponenten werden durch das Zweierkomplement dargestellt.

DEC_SIGN enthält das Vorzeichen der Zahl im KOI7-Code. Die möglichen Werte sind "+" (2BH) und "-" (2DH).

Funktion

mqcDECLOW_BIN konvertiert eine Dezimalzahl in eine Gleitkommazahl vom Typ BIN_TYPE und stellt sie unter BIN_PTR bereit. mqcDECLOW_BIN setzt voraus, daß die Dezimalzahl getrennt in Dezimalziffern, Dezimalexponent und Vorzeichen vorliegt (low-level-format).

Programmbeispiel

```
; Die EXTERN-Vereinbarung muß außerhalb der Segmentvereinbarungen
; stehen.

                EXTERN mqcDECLOW_BIN:FAR
;
                DSEG

BIN_PTR         DW      0,0             ; Adresse der Binärzahl
BIN_TYPE        DB      0               ; Datenart
DEC_LENGTH      DB      0               ; Anzahl der Dezimalziffern
DEC_PTR         DW      0,0             ; Adresse der Dezimalziffern
DEC_EXPONENT    DW      0               ; Dezimalexponent
DEC_SIGN        DB      0               ; Vorzeichen der Zahl
;
                .
                :
                CSEG
                .
                :
                MOV     DX,SEG BIN_PTR
                PUSH    DX
                MOV     DX,OFFSET BIN_PTR
                PUSH    DX
                CALLF   mqcDECLOW_BIN
                .
                :
```

4.8.4. Konvertierung long-real in temporary-real mqcLONG_TEMP

Parameter

LONG_REAL_PTR enthält die 2-Wort-Adresse der Gleitkommazahl der Datenart long-real.
TEMP_REAL_PTR enthält die 2-Wort-Adresse der Gleitkommazahl der Datenart temporary-real.

Funktion

Die Gleitkommazahl der Datenart long-real wird in die entsprechende Datenart temporary-real überführt.
NaNs, Infinities und minus Null bleiben erhalten.

Fehler

Der einzig mögliche Fehler tritt bei denormalisiertem Eingangswert auf. Bei unmaskiertem D-Exception wird der entsprechende Wert auf den Ausgabebereich gebracht. Unmaskierte D-Exceptions führen zum Aufruf des Exception-Handlers, nachden der Ausgabewert auf den Register-Stack gebracht wurde.

Programmbeispiel

```
          EXTERN  mqcLONG_TEMP
          .
          :
          DSEG
          .
          :
TEMP_NUM  DW      0,0,0,0,0
LONG_NUM  DW      0,0,0,0
          .
          :
          CSEG
          .
          :
          MOV     DX,SEG LONG_NUM      ; 2-Wort-Adresse von
          PUSH    DX
          MOV     DX,OFFSET LONG_NUM   ; LONG_NUM
          PUSH    DX
          MOV     DX,SEG TEMP_NUM      ; 2-Wort-Adresse von
          PUSH    DX
          MOV     DX,OFFSET TEMP_NUM   ; TEMP_NUM
          PUSH    DX
          CALLF   mqcLONG_TEMP
          .
          :
```

4.8.5. Konvertierung short-real in temporary-real
mqcSHORT_TEMP

Parameter

SHORT_REAL_PTR und TEMP_REAL_PTR enthalten die 2-Wort-Adressen der zu konvertierenden bzw. der konvertierten Zahl. Sie sind vor dem Aufruf des Konvertierungsprogramms auf den WM86-Stack zu bringen.

Funktion

Die Zahl in short-real wird durch Anfügen von Nullen auf temporary-real erweitert. NaNs, Infinities und minus Null werden in entsprechende Werte überführt.

Fehler

Ein Fehler (D-Exception) tritt nur bei denormalisiertem Eingangswert auf. Ist das Ereignis unmaskiert, wird der entsprechende Wert auf den Ausgabepuffer gebracht. Bei unmaskiertem D-Bit erfolgt ein Sprung zum Exception-Handler. Der Register-Stack enthält den denormalisierten Wert in der Datenert temporary-real.

Programmbeispiel

```
           EXTERN mqcSHORT_TEMP
           .
           :
           DSEG
           .
           :
SHORT_NUM  DW     0,0
TEMP_NUM   DW     0,0,0,0,0
           .
           :
           CSEG
           .
           :
           MOV    DX,SEG_SHORT_NUM     ; 2-Wort-Adresse von
           PUSH   DX
           MOV    DX,OFFSET SHORT_NUM  ; SHORT_NUM
           PUSH   DX
           MOV    DX,SEG TEMP_NUM      ; 2-Wort-Adresse von
           PUSH   DX
           MOV    DX,OFFSET TEMP_NUM   ; TEMP_NUM
           PUSH   DX
           CALLF  mqcSHORT_TEMP
           .
           :
```

4.8.6. Konvertierung temporary-real in long-real mqcTEMP_LONG

Parameter

TEMP_REAL_PTR und LONG_REAL_PTR sind die 2-Wort-Adressen der unkonvertierten bzw. der zu konvertierenden Zahlen. Sie sind vor Aufruf des Konvertierungsprogramms auf den WM86-Stack zu bringen.

Funktion

Die Zahl im Temporary-Real-Format wird durch Rundung in eine Long-Real-Zahl überführt. Liegt der Wert der zu konvertierenden Zahl genau zwischen zwei Long-Real-Werten, so wird derjenige gewählt, dessen letztes Bit gleich Null ist.

Fehler

Es können fünf verschiedene Fehler auftreten:

Overflow:
Liegt der Exponent außerhalb des für long-real gültigen Bereiches, wird im Ausgabepuffer ein vorzeichenbehaftetes Infinity und das Exception-Bit im Statuswort des WM87-Prozessors

gesetzt. Bei unmaskiertem Exception wird der Exception-Handler aufgerufen. Im Register-Stack steht der entsprechend long-real gerundete Eingangswert.

Underflow:
Im Ausgabepuffer steht das entsprechend long-real gerundete Ergebnis des Underflows. Bei unmaskiertem U-Exception erfolgt ein Sprung zum Exception-Handler. Der Register-Stack enthält die entsprechend long-real gerundete Mantisse und den unveränderten Exponenten der zu konvertierenden Zahl.

Genauigkeitsverlust:
Er tritt dann ein, wenn die zu konvertierende Zahl in den Positionen, die in long-real nicht berücksichtigt werden, von Null verschiedene Bits hat. Bei maskiertem P-Exception steht die gerundete Zahl im Ausgabebereich. Ist das P-Exception unmaskiert, wird der Exception-Handler aufgerufen. Die zu konvertierende Zahl steht im Register-Stack.

Unnormalisierter Eingangswert:
Er wird angezeigt, wenn der Eingangswert unnormalisiert ist, kein Underflow auftritt und die Rundung zu keinem normalisierten Wert führt. Das Ergebnis der Konvertierung ist dann ein Infinity. Bei unmaskiertem I-Exception steht bei Aufruf des Exception-Handlers der Eingangswert im Register-Stack.

Beschneidung von NaN:
Es werden nur diejenigen NaN im Zielformat exakt dargestellt, deren abgetrennte Bitpositonen keine gesetzten Bits enthalten. Bei allen anderen NaNs wird das I-Exception-Bit im Prozessorstatuswort gesetzt. Bei rückgesetzter I-Maske im Steuerwort steht bei Aufruf des Exception-Handlers der Eingangswert im Register-Stack. Wurde das I-Exception maskiert, dann wird der beschnittene Eingangswert zum Ausgabepuffer gebracht, wenn er ungleich Null ist. Ansonsten werden die Bitpositionen 5 bis 12 der Long-Real-Mantisse durch das höchstwertige Nicht-Null-Byte des Eingangswertes gebildet.

Programmbeispiel

```
            EXTERN mqcTEMP_LONG:FAR
            .
            :
            DSEG
            .
            :
TEMP_REAL   DW     0,0,0,0,0
LONG_REAL   DW     0,0,0,0
            .
            :
            CSEG
            .
            :
            PUSH   DS
            MOV    AX,OFFSET TEMP_REAL
            MOV    BX,OFFSET LONG_REAL
            PUSH   AX
            PUSH   DS
            PUSH   BX
            CALLF  mqcTEMP_LONG
            .
            :
```

4.8.7. Konvertierung temporary-real in short-real mqcTEMP_SHORT

Parameter

TEMP_REAL_PTR und SHORT_REAL_PTR enthalten die 2-Wort-Adressen der Gleitkommazahlen.

Funktion

Die Zahl in der Datenart temptorary-real wird auf eine entsprechende der Datenat short-real gerundet. Liegt der Wert der zu konvertierenden Zahl genau zwischen zwei Short-Real-Zahlen, so wird diejenige gewählt, deren letztes Bit gleich Null ist.

Fehler

Es können die gleichen Fehler wie bei mqcTEMP_LONG auftreten. Bezüglich der Fehlerreaktionen gelten die gleichen Festlegungen mit einer Ausnahme: Ergibt die Beschneidung eines NaN lediglich Nullen in den signifikanten Stellen, so werden die Bitpositionen 0 bis 7 der Short-Real-Zahl durch das höchstwertige Nicht-Null-Byte der Temporary-Real-Zahl gebildet.

Programmbeispiel

```
            EXTERN  mqcTEMP_SHORT:FAR
            .
            :
            DSEG
            .
            :
TEMP_NUM    DW      0,0,0,0,0
SHORT_NUM   DW      0,0,0
            .
            :
            CSEG
            .
            :
            MOV     AX,OFFSET TEMP_NUM
            MOV     BX,OFFSET SHORT_NUM
            PUSH    DS
            PUSH    AX
            PUSH    DS
            PUSH    BX
            CALLF   mqcTEMP_SHORT
            .
            :
```

5. Bibliothek für mathematische Funktionen

5.1. Vorbemerkungen

In diesem Abschnitt werden die Programme der CEL87 beschrieben. Diese Bibliothek realisiert mathematische Funktionen in den Datenformaten integer und Gleitkomma. Der WM87 bildet die Grundlage für die Berechnung dieser Funktionen. CEL87 liefert die programmäßige Unterstützung, die zur Realisierung weitgefächerter Anwenderforderungen nötig ist.
Im folgenden sind die Programme der CEL87 in gegliederten Funktionsgruppen aufgeführt. Das Präfix mqer dient zur Kenntlichmachung der Programmnamen und zur weitgehenden Vermeidung von Doppelvereinbarungen.

Rundungs- und Beschneidungsfunktionen:

mqerIEX rundet eine Gleitkommazahl zum nächsten Integer-Wert, bei gleichen Abständen auf den geraden Integer-Wert. Das Ergebnis ist eine Gleitkommazahl.

mqerIE2 entspricht mqerIEX mit 16 Bit integer.

mqerIE4 entspricht mqerIEX mit 32 Bit integer.

mqerIAX rundet eine Gleitkommazahl zum nächsten Integer-Wert, bei gleichen Abständen auf den von Null entfernteren Wert. Das Ergebnis ist eine Gleitkommazahl.

mqerIA2 entspricht mqerIAX mit 16 Bit integer.

mqerIA4 entspricht mqerIAX mit 32 Bit integer

mqerICX beschneidet die Gleitkommazahl auf den Integer-Wert. Das Ergebnis ist eine Gleitkommazahl.

mqerIC2 entspricht mqerICX mit 16 Bit integer.

mqerIC4 entspricht mqerICX mit 32 Bit integer.

Logarithmische und Exponentialfunktionen:

mqerLGD berechnet den Dezimallogarithmus (Basis 10).
mqerLGE berechnet den natürlichen Logarithmus (Basis e).
mqerEXP berechnet die Potenz e**x.
mqerY2X berechnet die Potenz y**x.
mqerYI2 berechnet die Potenz y**x mit Word-Integer-Exponenten.
mqerYI4 entspricht mqerYI2; der Exponent ist short-integer.
mqerYIS entspricht mqerYI2; der Exponent steht im WM86-Stack.

Trigonometrische und hyperbolische Funktionen:

mqerSIN, mqerCOS, mqerTAN
berechnen den Sinus, den Cosinus und den Tangens.
mqerASN, mqerACS, mqerATN
berechnen die zugehörigen inversen Funktionen.
mqerSNM, mqerCSM, mqerTNM
berechnen die zugehörigen hyperbolischen Funktionen.
mqerAT2 ist eine Spezialversion der Arcustangens-Funktion (ATN).

Weitere Funktionen:

mqerDIM ist eine Differenzfunktion für FORTRAN.
mqerMAX berechnet das Maximum zweier Gleitkommazahlen.
mqerMIN berechnet das Minimum zweier Gleitkommazahlen.
mqerSGH kombiniert das Vorzeichen einer Zahl mit der Mantisse einer anderen.
mqerMOD berechnet den Modulus einer Division. Das Vorzeichen entspricht dem des Dividenten.
mqerRMD berechnet den Modulus einer Division. Das Vorzeichen ist positiv.

5.2. Vereinbarung der CEL87-Programme in ASM86-Programmen

In jedem Quellprogrammodul, der CEL87-Prozeduren aufruft, sind die Namen als externe Symbole vom Typ FAR zu vereinbaren. Es ist günstig, diese Vereinbarung zu Beginn zu treffen und nur diejenigen Programme aufzuführen, die auch tatsächlich genutzt werden.

5.3. Stack-Nutzung

CEL87 benötigt 50 Byte im WM86-Stack. Wenn ein Programm der CEL87 durch ein anderes aufgerufen wird, das selbst CEL87-Routinen nutzt (z. B. der Exeption-Handler), sind zusätzlich 50 Byte erforderlich. Das gilt auch für alle weiteren rekursiven Aufrufe von CEL87-Funktionen.
In CEL87 ist ein 50-Byte-Stacksegment definiert, so daß die ersten 50 Byte automatisch durch LINK86 vereinbart werden.
CEL87 benötigt vier WM87-Stackpositionen zur Berechnung der Funktionswerte. Das heißt, daß die auf den Positionen ST0 und ST1 stehenden Eingangswerte bei Abschluß der Operation nicht mehr zur Verfügung stehen. Die Stackpositionen ST6 und ST7 müssen vor Aufruf eines Programms der CEL87 gelöscht sein. Tritt während der Berechnung Stacküberlauf auf, sind die Ergebnisse undefiniert.
Alle CEL87-Routinen sind wiederaufrufbar, d. h. sie können unterbrochen und durch ein anderes Programm aufgerufen werden. Das ist deshalb möglich, weil alle Arbeitszellen im WM86- bzw. WM87-Stack liegen. Es ist jedoch notwendig, vor Aufruf eines CEL87-Programms durch einen Exeption-Handler die Elemente des Register-Stack zu retten und bei Rückkehr wieder zu aktualisieren. Dabei sind alle 8 Stackpositionen zu berücksichtigen, weil die 4 tatsächlich benutzten nicht ermittelt werden können.

5.4. Registernutzung

Die Nutzung der WM86-Register durch CEL87-Programme entspricht den allgemeinen Festlegungen wonach die Inhalte der Register AX, BX, CX, DX, SI und ES zerstört werden können und die Inhalte der Register BP, SS und DS erhalten bleiben müssen. Das Stack-Pointer-Register SP wird nur durch die Funktion mqerYIS verändert, die einen Eingangsparameter im WM86-Stack erwartet. Alle CEL87-Programme retten das WM87-Steuerwort und aktualisieren es wieder vor Rückkehr ins aufrufende Programm. Basis der Berechnungen ist bei allen Funktionen außer mqerDIM die Datenart temporary-real. Der Rundungsmodus ist CEL87-spezifisch. mqerDIM nutzt die Genauigkeits- und Rundungsmodi, die bei Aufruf eingestellt sind.
Die Programme der CEL87 setzen die Exceptionbits im WM87-Statuswort nicht zurück. Ein vor oder während der Abarbeitung einer CEL87-Funktion gesetztes Exceptionbit wird bis zur Rückkehr ins rufende Programm nicht verändert. Dadurch ist es möglich, nach einer Reihe von Operationen festzustellen, ob ein bestimmtes maskiertes Exception aufgetreten ist.

5.5. Exceptions

Die Programme der CEL87 beinhalten den Fehleranzeigemechanismus des WM87. Wenn ein WM87-Fehler auftritt, wird das entsprechende Bit im Statuswort des WM87 gesetzt. Unter Nutzung des WM87-Befehlssatzes kann der Anwender über das Steuerwort die Reaktion auf bestimmte Exceptions festlegen. Mit Hilfe von unmaskierten Exceptions und Exeption-Handlern ist es möglich, Ausnahmesituationen zu erkennen und darauf zu reagieren, ohne daß nach jeder CEL87-Funktion eine Abfrage des Statuswortes erfolgen muß.

Es gibt lediglich sieben CEL87-Funktionen, bei denen das gesetzte P-Exception-Bit Bedeutung besitzt. Dabei handelt es sich um die Programme mqerIC2, mqerIC4, mqerICX, mqerIE2, mqerIE4, mqerIEX und mqerSGN. Bei allen anderen sollte das P-Bit maskiert sein.

Wenn ein unmaskiertes Exception durch eine CEL87-Funktion verursacht wurde, wird der Exception-Handler aufgerufen. Die Angabe, welche Stack-Register zu diesem Zeitpunkt genutzt wurden, ist in der Beschreibung der einzelnen Programme enthalten. Allgemein gelten für alle Funktionen bei Aufruf eines Exception-Handlers die folgenden Festlegungen:

- Die 20-Bit-Rückkehradresse des betreffenden CEL87-Programms ist im Operandenregister des WM87 enthalten.

- Das WM86-Statuswort enthält den bei Aufruf der Funktion innegehabten Wert.

- Das WM86-Befehlsregister ist auf OFFFFFH gesetzt.

- Das WM87-Operationsregister enthält einen funktionsspezifischen Wert. Er wird in der Beschreibung der einzelnen Programme genannt. In Anlage 16 sind alle Codes aufgelistet. Der Operationscode wird durch drei hexadezimale Zeichen dargestellt. Die erste Ziffer gibt die Anzahl der Positionen auf dem WM87-Stack an, die vor Auftreten des Trap durch die aktuelle CEL87-Funktion belegt wurden.

Die restlichen zwei Ziffern kennzeichnen das CEL87-Programm. Der Exception-Handler kann zunächst prüfen, ob der Trap durch eine CEL87-Operation hervorgerufen wurde. Das ist der Fall, wenn das WM87-Befehlsadreßregister auf FFFFFH gesetzt ist. Danach kann durch Auswertung des WM87-Operationscode-Registerinhalts die trapverursachende Funktion ermittelt werden.

5.6. Linken der Programme

Die Programme der mathematischen Funktionen sind in der Bibliothek CEL87.L86 enthalten. Sie nutzen entweder den Schaltkreis WM87 oder dessen Emulator EWM87. Dementsprechend sind die zugehörigen Interfacebibliotheken WM87.L86 bzw. EWM87.L86 beim Linken anzugeben.

Werden Programme der Exception-Handler-Bibliothek genutzt, so muß deren Name EH87.L86 im LINK86-Kommando nach CEL87.L86 stehen. Die Reihenfolge der Objektprogramme muß wie folgt lauten:

Anwenderprogramm
DCON87.L86 (wenn genutzt)
CEL87.L86
EH87.L86 (wenn genutzt)
WM87.L86 (wenn Schaltkreis genutzt)
EWM87, EWM87.L86 (wenn Emulator genutzt)

5.7. Beschreibung der Programme

5.7.1. Programm zur Berechnung des Arcuscosinus mqerACS

Eingangsparameter

Der Eingangswert steht auf der Register-Stack-Position.

Funktion

mqerACS liefert den Wert im Bogenmaß, dessen Cosinus gleich dem Eingangswert ist. Die Ergebnisse liegen im Bereich 0 bis Pi. Zulässige Eingangswerte liegen zwischen -1 und +1 (einschließlich). Null, Pseudonull und denormalisierte Eingangswerte liefern Pi/2. Auch unnormalisierte Werte, die nicht kleiner als 2^{-63} sind, ergeben Pi/2.

Ausgangsparameter

Der Ausgangswert steht auf der Register-Stack-Position des Eingangswertes.

Fehler

Da die Werte des Cosinus im Bereich $-1 \leq x \leq 1$ liegen, liefern alle anderen Eingangswerte invalid operation. Ebenso bewirken Infinities, NaNs und unnormalisierte Werte, die kleiner als 2^{-63} sind, das Setzen des I-Bit im WM87-Statuswort.
Ist das I-Bit unmaskiert, steht bei Aufruf des Exception-Handlers der Eingangswert auf dem WM87-Stack und das WM87-Operationscoderegister enthält den Wert 175H.
Bei maskiertem Exception bleiben NaNs als Eingangswerte erhalten. Alle anderen unerlaubten Eingangswerte führen zu Indefinite.

Programmbeispiel

```
; Die EXTERN-Vereinbarung muß außerhalb aller Segmentdefinitionen
; stehen.
                 EXTERN mqerACS:FAR

                 CSEG
HYPOTHENUSE      RS      8      ; längste Seite eines rechtwinkligen
                                ; Dreiecks
ADJACENT_SIDE    RS      8      ; Seite am zu berechnenden Winkel
THETA_RADIANS    RS      8
THEDA_DEGREES    RS      8
RAD_TO_DEG       RS      10     ; mit der Konstanten 180/Pi zu
                                ; versorgen
;
; Die nachfolgenden Befehle berechnen den Winkel des Dreiecks.
; Es wird vorausgesetzt, daß die benötigten Werte
; bereitgestellt werden.
;
                 DSEG
                 .
                 :
                 FLD64  ADJACENT_SIDE ; (x) = ADJACENT_SIDE
                 FDIV64 HYPOTHENUSE   ; (x) = ADJACENT_SIDE /
                                      ;       HYPOTHENUSE
                 CALLF  mqerACS       ; Bogenmaß ist in (x)
                 FST64  THETA_RADIANS ; Ergebnis abspeichern
                 FID80  RAD_TO_DEG
                 FMUL                 ; (x) wird in Grad konvertiert
                 FST64P THETA_DEGREES ; Ergebnis in Grad abspeichern
                 .
                 :
```

5.7.2. Programm zur Berechnung des Arcussinus mqerASN

Eingangsparameter

Der Eingangswert steht auf der aktuellen Register-Stack-Position.

Funktion

mqerASN liefert den Wert im Bogenmaß, dessen Sinus gleich dem Eingangswert ist. Die Ergebnisse liegen im Bereich -Pi/2 bis +Pi/2.

Ausgangsparameter

Das Ergebnis steht auf der WM87-Stack-Position des Eingangswertes.

Fehler

Alle Eingangswerte, die außerhalb des Bereiches -1 bis +1 (einschließlich) liegen, alle NaNs und Infinities bewirken das Setzen des I-Bits. Außerdem führen unnormalisierte Zahlen kleiner als 2^{-63} zu diesem Fehler.

Ist das I-Bit im Steuerwort des WM87 rückgesetzt (unmaskiert), dann steht bei Aufruf des Exception-Handlers der Eingangswert im Register-Stack. Das Operationscoderegister enthält den Wert 174H.
Bei maskiertem Exception werden eingangsseitige NaNs nicht verändert. Alle anderen unerlaubten Eingangswerte führen zu Indefinite als Ergebnis.

Programmbeispiel

```
; Die EXTERN-Vereinbarung muß außerhalb aller Segment-
; definitionen stehen.
;
                EXTERN mqerASN:FAR
                .
                :
                DSEG
HYPOTHENUSE     RS      8               ; Hypothenuse
OPPOSITE_SIDE   RS      8               ; Seite am zu berechnenden
;                                         Winkel
THETA_DEGREES   RS      8
RAD_TO_DEG      RS      10              ; Konstante 180/Pi
                .
                :
                CSEG
                .
                :
; Es wird vorausgesetzt, daß die benötigten Werte bereits
; bereitgestellt wurden.
;
                FLD64  OPPOSITE_SIDE ; (x) = OPPOSITE_SIDE
                FDIV64 HYPOTHENUSE   ; (x) = OPPOSITE_SIDE /
;                                            HYPOTHENUSE
                CALLF  mqerASN       ; x enthält den Winkel
;                                      im Bogenmaß
                FLD80  RAD_TO_DEG
                FMUL
                FST64P THETA_DEGREES
```

5.7.3. Programm zum Berechnen des Arcustangens mqerAT2

Eingangsparameter

Der Eingangswert x steht auf der aktuellen Register-Stack-Position, y auf der aktuellen Position + 1.

Funktion

mqerAT2 führt einen Teil der Konvertierung von einem rechtwinkligen in ein Polar-Koordinatensystem durch. Wenn die Eingangswerte x und y Koordinaten eines Punktes im x-y-Koordinatensystem sind, berechnet mqerAT2 den Winkel im Bogenmaß, den eine Gerade durch diesen Punkt mit der positiven x-Achse bildet. Der Winkel kann sich von Pi unterhalb der x-Achse bis Pi von x = 0 oberhalb der x-Achse erstrecken.

Die Berechnung erfolgt nach folgenden Richtlinien:

für x > 0 : arctangent(y/x)
für x = 0 : mqerSGN(Pi/2,y)
für x < 0 : arctangent(y/x) + mqerSGN(Pi,y)

Es ist zulässig, daß einer der Eingangswerte ein Infinity ist. Der Punkt wird dann als unendlich weit entfernt auf dieser Achse angenommen. Das Ergebnis ist der Winkel, den diese Achse mit der positiven x-Achse bildet. Für Punkte nahe der x-Achse enthält der Ergebniswinkel das Vorzeichen von y. Demzufolge sind folgende Kombinationen möglich:

x = +Infinity : Ergebnis = 0 mit dem Vorzeichen von y.
x = -Infinity : Ergebnis = Pi mit dem Vorzeichen von y.
y = +Infinity : Ergebnis = +Pi/2.
y = -Infinity : Ergebnis = -Pi/2.

In all diesen Fällen ist es erlaubt, daß der von Infinity verschiedene Wert unnormalisiert ist. Es ist zu beachten, daß auch im Projektiv-Modus zwischen + und -Infinity unterschieden wird.
mqerAT2 verarbeitet denormalisierte Eingangswerte. Seine Reaktion ist abhängig davon, ob das D-Bit im Steuerwort gesetzt ist oder nicht. Befindet sich der WM87 im Normalisierungsmodus (D ist unmaskiert), dann werden alle denormalisierten Werte auf Null gesetzt. Im Warning-Modus (D ist maskiert) werden die denormalisierten Werte durch unnormalisierte mit dem gleichen numerischen Wert ersetzt.
mqerAT2 prüft lediglich das D-Maskenbit. Es setzt weder das D-Bit im Statuswort noch ruft es den Exception-Handler auf.
In einigen Fällen erfolgt bei einem unnormalisierten Eingangswert keine Fehlerreaktion. Voraussetzung dafür ist, daß durch x und y ein Punkt beschrieben wird, der nahe genug an einer der Achsen liegt. Als genügend nah wird angesehen, wenn der Quotient x/y bzw. y/x kleiner als 2^{-63} ist. Wenn der Punkt nahe der positiven x-Achse liegt, stellt das Verhältnis y/x mit dem Vorzeichen von y das Ergebnis dar. Liegt der Punkt in der Nähe der anderen drei Achsen liefert mqerAT2 den entsprechenden Winkel zur positiven x-Achse im Bogenmaß als Resultat:

- Pi mit dem Vorzeichen von y für die negative x-Achse
- Pi/2 für die positive y-Achse
- -Pi/2 für die negative y-Achse

Eine Fehlerreaktion erfolgt in jedem Falle, wenn beide Eingangswerte unnormalisiert sind.

Ausgangsparameter

Das Ergebnis steht auf der Register-Stack-Position des zweiten Eingangswertes. Der WM87-Stack-Pointer wurde um 1 vermindert.

Fehler

Das I-Bit im Statuswort wird gesetzt, wenn ein oder beide Eingangswerte NaNs sind. Ist das I-Exception maskiert, stellt das NaN das Ergebnis dar. Sind beide Eingangswerte NaN, dann wird der größere Wert genutzt.
Das I-Bit wird ebenfalls gesetzt, wenn beide Eingangswerte 0 sind. Der Punkt liegt danach im Schnittpunkt der x-y-Achsen. Ein Winkel kann nicht definiert werden.
Unnormalisierte Eingangswerte, die nicht den oben genannten

Bedingungen entsprechen, führen ebenfalls zu invalid operation. In all diesen Fällen wird mqerAT2 bei maskiertem I-Exception Indefinite als Resultat liefern.
Bei allen unmaskierten I-Exceptions wird der Exoeptin-Handler aufgerufen. Die unveränderten Eingangswerte stehen auf dem WM87-Stack. Das Operationscoderegister enthält 277H.
Wenn die Mantisse des Ergebnisses zu klein ist, um in temporary-real dargestellt werden zu können, wird das Underflow-Bit gesetzt. Bei gesetztem U-Bit im Steuerwort (maskiertes Exception) liefert mqerAT2 wenn möglich einen denormalisierten Wert als Ergebnis, ansonsten Null.
War der Underflow unmaskiert, dann wird der Exception-Handler aufgerufen. Im Operationscoderegister steht 277H. Das Ergebnis wird durch die Mantisse in Verbindung mit einem modifizierten Exponenten dargestellt. Der wahre Exponent ergibt sich aus dem modifizierten durch Subtraktion von 24576.

Programmbeispiel

```
;
EXTERN        mqerAT2:FAR
              :
              DSEG
              :
POALR_THETA   RS      8
POALR_R       RS      8
REC_C         RS      8              ; x-Koordinate
REC_Y         RS      8              ; y-Koordinate
              :
;
;
; Es wird vorausgesetzt, daß die benötogten Werte bereits
; bereitgestellt wurden.
;
              :
              FLD64    REC_y         ; y-Koordinate zum Reg.-Stack
              FLD64    REC_X         ; x-Koordinate zum Reg.-Stack
              CALLF    mqerAT2       ; Berechnung des Winkels
              FST64P   POLAR_THETA   ; abspeichern
              FLD64    REC_Y         ; y zum Register-Stack
              FMUL     ST,ST         ; y**2
              FLD64    REC_X         ; x zum Register-Stack
              FMUL     ST,ST         ; x ** 2
              FADD64P  ST1,St        ; Summe
              FSQRT                  ; Radius
              FST64P   POLAR_R       ;
              :
```

5.7.4. Programm zur Berechnung des Arcustangens mqerATN

Eingangsparameter

Der Eingangswert steht auf der aktuellen Register-Stack-Position.

Funktion

mqerATN liefert den Wert im Bogenmaß, dessen Tangens gleich dem Eingangswert ist. Das Ergebnis liegt zwischen -Pi/2 und +Pi/2. Nullen, Pseudonullen, denormalisierte Werte und unnormalisierte Werte kleiner 2^{-63} bilden unverändert das Resultat.
Infinities sind als Eingangswerte erlaubt, unabhängig davon, in welchem Infinity-Modus sich der WM87 befindet. Negative Infinities liefern als Ergebnis -Pi/2. Positive Infinities führen zu +Pi/2.

Ausgangsparameter

Der Ausgangswert steht auf der Register-Stack-Position des Eingangswertes.

Fehler

NaNs und unnormalisierte Werte, die größer als 2^{-63} sind, führen zum I-Exception. Bei maskiertem Exceptionbit im Steuerwort haben unnormalisierte Eingangswerte Indefinities zum Ergebnis. Eingabenseitige NaNs werden zum Resultat.
Ist das Ergebnis unmaskiert, bleiben die Eingangsparameter bei Aufruf des Exception-Handlers erhalten. Im Operationscoderegister steht der Wert 176H.

Programmbeispiel

```
;
                EXTRN  mqerATN: FAR
                .
                :
                DSEG
OPPOSITE_SIDE   RS     8            ; Seite gegenüber dem Winkel
ADJACENT_SIDE   RS     8            ; Seite am Winkel
THETA_RADIANS   RS     8
                CSEG
;
                .
                :
; Es wird vorausgesetzt, daß die benötigten Werte
; bereits bereitgestellt wurden
                FLD64  OPPOSITE_SIDE ; (x) = OPPOSITE_SIDE
                FDIV64 ADJACENT_SIDE ; (x) = OPPOSITE_SIDE /
;                                        ADJACENT_SIDE
                CALLF  mqerATN       ;
                FST64P THETA_RADIANS ;
```

5.7.5. Programm zur Berechnung des Cosinus mqerCOS

Eingangsparameter

Der Eingangswert steht auf der aktuellen Stack-Position des WM87.

Funktion

mqerCOS berechnet den Cosinus des Eingangswertes im Bogenmaß. Folgende Werte liefern als Ergebnis 1:

- Null
- Pseudonull
- denormalisierte Werte
- unnormalisierte Werte < 2^{-63}

Ausgangsparameter

Der berechnete Cosinus steht auf der Register-Stack-Position des Eingangswertes.

Fehler

Das I-Exceptionbit wird bei NaN, Infinities und unnormalisierten Werten, die größer als 2^{-63} sind, gesetzt.
Wurde das I-Exception maskiert, wird ein NaN als Eingangswert zum Ergebnis. Alle anderen unzulässigen Eingangswerte führen zu Indefinite.
Ist das Exception unmaskiert, bleiben die Eingangsparameter erhalten. Der Exception-Handler wird aufgerufen. Im Operationscoderegister steht 172H.

Programmbeispiel

```
;
                EXTRN  mqerCOS: FAR
                .
                :
                DSEG
;
POLAR_TMETA     RS     8
POLAR_R         RS     8
REC_X           RS     8
DEG_TO_RAD      RS     10                ; die Konstante Pi/180
                .
                :
                CSEG
                .
                :
; Berechnung der x-Koordinate im x/y-Koordinatensystem
; ausgehend vom Polar-Koordinatensystem.
; Es wird vorausgesetzt, daß die benötigten Werte
; zuvor bereitgestellt wurden
                .
                :
                FLD64  POLAR_TMETA       ; Winkel im Grad
                FLD80  DEG_TO_RAD
                FMUL                     ; Winkel im Bogenmaß
                CALLF  mqerCOS
                FMUL64 POLAR_R           ; x - Koordinate
```

5.7.6. Programm zur Berechnung des Cosinus Hyperbolicus mqerCSH

Eingangsparameter

Der Eingangswert steht auf der aktuellen Register-Stack-Position.

Funktion

mqerCSH berechnet den hyperbolischen Cosinus von x, wobei x ein Winkel im Bogenmaß ist. Nullen, Pseudonullen, denormalisierte Werte und unnormalisierte Werte, die kleiner als 2^{-63} sind, ergeben 1. Infinities bleiben erhalten. Es erfolgt keine Fehleranzeige.

Ausgangsparameter

Das Ergebnis steht auf der Register-Stack-Position des Eingangswertes.

Fehler

Bei NaN und unnormalisierten Werten, die größer 2^{-63} sind, wird das I-Bit gesetzt.
Ist das I-Bit unmaskiert, steht bei Aufruf des Exception-Handlers der Eingangswert im WM87-Stack. Im anderen Falle bleibt ein NaN erhalten, unnormalisierte Werte < 2^{-63} ergeben Indefinit.
mqerCSH erzeugt einen Overflow, wenn der Eingangswert größer als 11355 oder kleiner als -11355 ist. Bei unmaskiertem O-Exception wird der Exception-Handler aufgerufen. Der Eingangswert steht noch im WM87-Stack. Ist das O-Bit im Steuerwort gesetzt (maskiertes Exception), ist das Ergebnis Indefinite.
Bei Aufruf des Exception-Handlers steht im Operationscoderegister 16FH.

Programmbeispiel

```
;
                EXTRN   mqerCSH : FAR
                .
                :
                DSEG
                .
                :
INPUT_VALUE     RS      8
OUTPUT_VALUE    RS      8
                .
                :
; Es wird vorausgesetzt, daß die benötigten Werte
; zuvor bereitgestellt wurden.
                .
                :
                CSEG
                FLD64   INPUT_VALUE
                CALLF   mqerCSH
                FST64P  OUTPUT_VALUE
                .
                :
```

5.7.7. Programm zur Bildung der positiven Differenz mqerDIM

Eingangsparameter

x bzw. y stehen auf der aktuellen Register-Stack-Position, bzw. auf der um 1 erhöhten.

Funktion

mqerDIM vergleicht zunächst die beiden Eingangswerte. Ist x größer als y, dann ist das Ergebnis 0. Ist y größer als x, dann ist die Differenz (y-x) das Resultat.
Die Eingangswerte können Infinities sein, wenn der Affin-Modus (vorzeichenbehaftete Infinities) gesetzt ist und wenn die Werte nicht identisch sind.
Gemäß den obigen Festlegungen gelten folgende Regeln:

x	y	Resultat
+Infinity		+0
	-Infinity	+0
-Infinity		+Infinity
	+Infinity	+Infinity

mqerDIM nutzt die bei Aufruf eingestellten Genauigkeits- und Rundungsmodi. Entsprechend können die Ergebnisse in Abhängigkeit von den Bits im Steuerwort variieren.

Ausgangsparameter

Die aktuelle Register-Stack-Position ist um 1 erhöht. Sie enthält das Resultat.

Fehler

Das Programm meldet I-, O-, U- und D-Exceptions. Das D-Bit wird bei un- und denormalisierten Eingangswerten gesetzt. Im Warning-Modus (D maskiert) wird die Berechnung fortgesetzt. Im Normalisierungsmodus (D unmaskiert) wird der Exception-Handler aufgerufen. Das Operationscoderegister enthält den Funktionscode der Operation, bei der das D-Exception auftrat (entweder FCOM oder FSUB). Die Eingangswerte stehen im Register-Stack. Die meisten Exception-Handler werden die Normalisierung durchführen, die unterbrochene Operation (FCOM bzw. FSUB) wiederholen und in mqerDIM fortsetzen. Der entsprechende Exception-Handler der Fehlerbehandlungsbibliothek EH87 realisiert diese Maßnahmen.
Eingangsseitige NaN bewirken das Setzen des I-Bit. Bei maskiertem I-Exception bildet das NaN das Resultat. Sind beide Eingangswerte NaN, wird der größere Wert genutzt.
Das I-Bit im Statuswort wird auch gesetzt, wenn der Prozessor im Projektiv-Modus (vorzeichenlose Infinities) arbeitet, und im Eingangswert im Infinity ist oder wenn der Affin-Modus eingestellt wurde und beide Eingangswerte die gleichen Infinities sind. In diesen Fällen führt das maskierte Exception zu Indefinite als Ergebnis.
Alle I-Exceptions führen bei rückgesetztem I-Bit im Steuerwort (Exception unmaskiert) zum Aufruf des Exception-Handlers. Eingangsparameter sind unverändert. Im Operationscoderegister des

Prozessors steht 265H.
Wenn sich das Ergebnis aus der Differenz (y-x) ergibt, kann bei der Subtraktion ein Overflow bzw. ein Underflow auftreten.
Das maskierte O-Exception führt zu +Infinity als Ergebnis. Das maskierte U-Exception hat einen denormalisierten Wert zur Folge. Das Ergebnis ist 0, wenn die Bildung eines gradual denormalisierten Wertes nicht möglich ist.
Bei unmaskiertem O- bzw. U-Exception wird der Exception-Handler gerufen. Im Operationscoderegister steht 165H. Anstelle der Eingangswerte enthält der Register-Stack das Ergebnis mit modifizierten Exponenten. Der exakte Exponent ergibt sich nach einem Underflow durch Subtraktion, bei einen Overflow durch Addition von 24576 zum modifizierten Exponenten.

Programmbeispiel

```
;
                 EXTRN  mqerDIM : FAR
                 .
                 :
                 DSEG
                 .
                 :
COSTS            RS     8
RECEIPTS         RS     8
                 .
                 :
PROFIT           RS     8
LOSS             RS     8
                 .
                 :
                 CSEG
                 .
                 :
; Es wird vorausgesetzt, daß die benötigten Werte
; schon bereitgestellt wurden
;
                 FLD64  RECEIPTS       ; Einnahmen
                 FLD64  COSTS          ; Kosten
                 CALLF  mqerDIM
                 FST64P PROFIT         ; Gewinn
;
                 FLD64  COSTS
                 FLD64  RECEIPTS
                 CALLF  mqerDIM
                 FST64P LOSS           ; Verluste
```

5.7.8. Programm zur Berechnung einer Potenz mqerEXP

Eingangsparameter

Der Exponent steht auf der aktuellen Position des Register-Stack.

Funktion

mqerEXP potenziert die Konstante e (2 718 281 828 499) mit dem Eingangswert x. e^x ist eine Standardfunktion, die von anderen Funktionen, z. B. der hyperbolischen, genutzt wird. Es sind sowohl positive als auch negative Werte des Exponenten zulässig.

Nullen, Pseudonullen denormalisierte und unnormalisierte Werte kleiner 2^{-63} ergeben 1.
Im Affin-Modus (vorzeichenbehaftete Infinities) sind Infinities als Eingangswerte erlaubt. Negative Infinities ergeben Null. Positive Infinities bilden unverändert das Resultat.

Ausgangsparameter

Der Wert der Potenz steht auf der Register-Stack-Position des Eingangswertes.

Fehler

NaNs und unnormalisierte Werte größer 2^{-63} bewirken das Setzen des I-Bit im Statuswort. Außerdem führen im Projektiv-Modus sowohl der positive als auch der negative Infinity zum I-Exception.
Wurde das I-Bit maskiert, dann führen alle Eingangswerte außer NaN zu Indefinit. NaNs bleiben erhalten.
Bei Aufruf des Exception-Handlers (Exception unmaskiert) steht noch der Eingangswert im Register-Stack.
Die größte darstellbare Zahl in der Datenart temporary-real ist etwa e^{11357}. Entsprechend bewirken Eingangswerte, die größer sind als 11357, einen Overflow, der im unmaskierten Fall +Infinity als Resultat erzeugt. Demgegenüber führen Exponenten, die kleiner als -11355 sind, zu einem Underflow. Dieser bewirkt, wenn U maskiert ist, die Bildung eines denormalisierten Wertes oder, wenn das nicht möglich ist, von Null als Ergebnis.
Sind U- und O-Exceptions unmaskiert, wird der Exception-Handler erreicht. Die Eingangswerte stehen unverändert im Register-Stack.
In jedem Falle enthält das Operationscoderegister bei Aufruf des Exception-Handlers 16BH.

Programmbeispiel

```
;
                EXTRN   mqerEXP:FAR
                :
                DSEG
                :
MINUS_2         RS      8
X_VALUE         RS      8
Y_VALUE         RS      8
NORM_CONSTANT   RS      10              ; Konstante 1/sqrt(2*PI)
                .
                :
                CSEG
                .
                :
; Berechnung der Normalverteilung
; Es wird vorausgesetzt, das die erforderlichen
; Werte schon bereitgestellt wurden.
;
                FLD64   X_VALUE
                FMUL    ST,ST
                FDN64   MINUS_2         ; Division durch -2.
                CALLF   mqerEXP
                FLD80   NORM_CONSTANT   ; Normalkonstante
                FMUL
;                                         Division durch Quad.-wurzel
;                                         von 2 * PI
                FST64P  Y_VALUE         ;
```

5.7.9. Programm zum Runden real in word-integer mqerIA2

Eingangsparameter

Die zu rundende Zahl steht auf der aktuellen Register-Stack-Position.

Funktion

Nichtnormalisierte Zahlen werden zunächst normalisiert. Danach erfolgt die Rundung zur nächstgelegenen Integer-Zahl. Liegt der Wert genau zwischen zwei Integer-Zahlen (gebrochener Teil der Zahl gleich .5), dann wird die von Null weiter entfernte genutzt. Negative Integer-Zahlen werden im Zweierkomplement dargestellt.

Beispiele

Gleitkommazahl	Integer-Zahl
3.1	0003
10.5	000B (11)
-2.5	FFFD (-3)

Ausgangsparameter

Das AX-Register enthält die Integer-Zahl. Der Eingangswert steht nicht mehr im Register-Stack.

Fehler

Der Wertebereich der durch mqerIA2 konvertierbaren Zahlen erstreckt sich von -32767.4... bis +32767.4... . Alle Werte, die außerhalb dieses Bereiches liegen, führen zu invalid operation (I-Exception). Außerdem bewirken NaNs und Infinities das Setzen des I-Bit.
Bei maskiertem I-Exception ist Integer- Indefinite 8000H das Ergebnis. Im anderen Fall wird der Exception-Handler aufgerufen. Das Operationscoderegister enthält 17EH. Der Eingangswert steht im Register-Stack.

Programmbeispiel

```
;
                EXTRN   mqerIA2:FAR
                .
                :
                DSEG
INTEGER_VAR     DW      0
REAL_VAR        RS      8
                .
                :
                CSEG
                .
                :
; Es wird vorausgesetzt, daß der Eingangswert
; bereitgestellt wurde.
                .
                :
                FLD64   REAL_VAR
                CALLF   mqerIA2
                MOV     INTEGER_VAR,AX
                .
                :
```

5.7.10. Programm zum Runden real in short-integer mqerIA4

Eingangsparameter

Die zu konvertierende Gleitkommazahl steht auf der aktuellen Position des Register-Stack.

Funktion

Zunächst werden unnormalisierte Zahlen normalisiert. Danach erfolgt die Konvertierung der Gleitkommazahl in eine Integer-Zahl durch Runden. Dabei wird der Gleitkommazahl die nächstgelegene Integer-Zahl zugeordnet. Bei gleichen Abständen wird die von Null entfernte Integer-Zahl genutzt. Das Ergebnis ist eine Integer-Zahl der Datenart short-integer.

Beispiel

Gleitkommazahl	Integer-Zahl
3.1	00000003H
10.5	0000000BH

```
    -2.5            FFFFFFFDH
```

Ausgangsparameter

Die Integer-Zahl steht im DX- und AX-Register, wobei DH das höchstwertige und AL das niederwertigste Byte enthalten.

Fehler

Der Wertebereich der durch mqerIA4 konvertierbaren Zahlen erstreckt sich von -2147483648.5 bis +2147483647.5 (ausschließlich). Zahlen außerhalb dieses Bereiches, Infinities und NaNs erzeugen einen I-Fehler. Wurde das I-Exception maskiert, dann ist 80000000H (integer-indefinit) das Resultat.
Bei Aufruf des Exception-Handlers (Exception unmaskiert) steht der Eingangswert im Register-Stack. Das Operationscoderegister enthält 168H.

Programmbeispiel

```
;
                EXTRN   mqerIA4:FAR
                .
                :
                DSEG
                .
                :
COUNT           DW      2
REAL_COUNT      RS      8
                .
                :
                CSEG
                .
                :
;
; Es wird vorausgesetzt, daß ein gültiger Wert
; in REAL_COUNT steht.
;
                FLD64   REAL_COUNT
                CALLF   mqerIA4
                MOV     BX, OFFSET COUNT
                MOV     WORD PTR [BX],AX; niederwertiges Wort
                INC     BX
                INC     BX
                MOV     WORD PTR [BX],DX; höherwertiges Wort
```

5.7.11. Programm zum Runden real in integer mqerIAX

Eingangsparameter

Die zu rundende Gleitkommazahl steht auf der aktuellen Register-Stack-Position.

Funktion

Nichtnormalisierte Werte werden vor dem Runden normalisiert. Die Gleitkommazahl wird anschließend in die nächstliegende Gleitkommazahl ohne gebrochenen Teil überführt. Sind die Abstände zu den

zwei benachbarten Werten gleich, wird derjenige genutzt, dessen Differenz mit Null verglichen, größer ist.
Infinities bleiben erhalten. Es folgt keine Fehleranzeige.

Ausgangswert

Die genannte Gleitkommazahl steht im Register-Stack.

Fehler

Das I-Exception wird angezeigt, wenn es sich bei dem zu rundenden Wert um ein NaN handelt. Bei maskiertem I-Exception bleibt das NaN erhalten. Im unmaskierten Falle erfolgt der Aufruf des Exception-Handlers. Im Operationscoderegister steht 167H.

Programmbeispiel

```
;
                 .
                 :
                 EXTRN   mqerIAX:FAR
                 .
                 :
                 DSEG
HOURS            RS      8         ; Testwert in Stunden
MINUTES          RS      8
SIXTY            DW      2         ; 60.0
                 .
                 :
                 CSEG
                 .
                 :
; Es wird vorausgesetzt, daß HOURS und SIXTY
; bereits mit entsprechenden Werten versorgt wurden
;
                 FLD64   HOURS     ; Testwert auf Register-Stack
                 FMUL32  SIXTY     ; Minuten als Gleitkommawert
                 CALLF   mqerIAX   ; Rundung auf Integer-Wert
                 FST64P  MINUTES
```

5.7.12. Programm zur Bildung von word-integer aus real mqerIC2

Eingangsparameter

Die zu rundende Gleitkommazahl steht auf der aktuellen Register-Stack-Position.

Funktion

Unnormalisierte Eingangswerte werden zunächst normalisiert. Ist x bereits ein Integer-Wert, dann bleibt er unverändert. Ansonsten wird der fraktionelle Teil der Gleitkommazahl abgetrennt. Auf diese Weise wird der Gleitkommazahl die betragsmäßig kleinere Integer-Zahl zugeordnet. Negative Zahlen werden im Zweierkomplement dargestellt.

Beispiel

real	integer
4	0004H
11.7	000BH
-6.9	FFFAH

Ausgangsparameter

Die zugehörige Integer-Zahl steht im AX-Register.

Fehler

Es können nur Zahlen innerhalb des Bereiches von -32769 bis +37768 in Integer-Zahlen einfacher Wortlänge überführt werden. Zahlen außerhalb dieses Bereiches, NaNs und Infinities verursachen das Exception invalid operation. Wurde das Exception maskiert, liefert das Programm das Integer-Indefinity 8000H als Ergebnis. Es ist zu beachten, daß auch ein zulässiger Eingangswert zu diesem Resultat führen kann. Eine eindeutige Aussage liefert nur das I-Bit im Statuswort des Prozessors.

Bei unmaskiertem I-Exception erfolgt ein Aufruf des Exception-Handlers. Der Eingangsparameter steht dabei auf der aktuellen Register-Stack-Position. Wenn eine Beschneidung der Real-Zahl stattgefunden hat, wird das P-Bit (Precision-Bit) gesetzt. Im maskierten Fall wird die gewünschte Integer-Zahl im AX-Register übergeben. Es wird empfohlen, das P-Exception zu maskieren, da ansonsten der Inhalt von AX zerstört werden kann, bevor ihn der Exception-Handler nutzt.
Bei Aufruf des Exception-Handlers steht im Operationscoderegister des WM87 der Wert 17EH.

Programmbeispiel

```
;
                    EXTRN  mqerIC2:FAR
                    .
                    :
                    DSEG
REAL_INPUT          RS     8
CONTROL_SETTING     DW     0
                    .
                    :
                    CSEG
;
; Es wird vorausgesetzt, daß REAL_INPUT einen
; gültigen Wert enthält.
;
                    FLD64  REAL_INPUT
                    CALLF  mqerIC2
                    MOV    CONTROL_SETTING,AX
                    .
                    :
```

5.7.13. Programm zur Bildung von short-integer aus real mqerIC4

Eingangsparameter

Die zu rundende Gleitkommazahl steht auf der aktuellen Register-Stack-Position.

Funktion

Nichtnormalisierte Zahlen werden vor dem Runden normalisiert. Ist x bereits ein Integer-Wert, dann bleibt er unverändert. Ansonsten wird der gebrochene Teil der Gleitkommazahl abgetrennt. Auf diese Weise wird der Real-Zahl die betragsmäßig kleinere Integer-Zahl zugeordnet. Negative Zahlen im Zweierkomplement dargestellt.

Ausgangsparameter

Die Integer-Zahl vom Typ short-integer steht in den Registern DX und AX. Dabei enthält DH das höchstwertigste und AL das niederwertigste Byte.

Fehler

Es werden nur Zahlen innerhalb des Bereiches

$-2147483649. < x < +2147483648.$

in Integer-Zahlen überführt. Real-Zahlen außerhalb dieses Bereiches, NaNs und Infinities verursachen das I-Exception. Wurde das Exception maskiert, liefert das Programm das Indefinity 80 000 000H als Ergebnis. Es ist zu beachten, daß auch ein zulässiger Eingangswert zu diesem Resultat führen kann. Eine eindeutige Aussage liefert nur das I-Bit im Statuswort des Prozessors.

Bei unmaskiertem I-Exception erfolgt ein Aufruf des Exception-Handlers. Dabei steht der Eingangsparameter auf der aktuellen Register-Stack-Position.

Wenn eine Beschneidung der Real-Zahl stattgefunden hat, wird das P-Bit (Precision-Bit) im Statuswort gesetzt. Im maskierten Fall wird die geforderte Integer-Zahl in den Registern DX und AX übergeben. Bei unmaskiertem P-Bit erfolgt der Aufruf des Exception-Handlers. Wurde der Exception-Handler aufgerufen, so enthält das Operationsregister in jedem Falle 179H.

Programmbeispiel

```
                    .
                    :
                    EXTRN   mqerIC4:FAR
                    .
                    :
                    DSEG
POPULATION          RS      8              ; Anzahl der Wähler
SUPPORT_SHARE       RS      8              ; Prozent der Zustimmungen
SUPPORT_VORES       RS      8              ; Anzahl der Zustimmung
                    .
                    :
                    CSEG
                    .
                    :
; Im folgenden wird die Anzahl der JA-Stimmen aus
; der Gesamtanzahl der Wähler und der Prozentzahl
; der JA-Stimmen berechnet. Es wird vorausgesetzt,
; daß POPULATION und SUPPORT_SHARE bereits
; initialisiert wurden.
                    .
                    :
                    FELD64  POPULATION
                    FMUL64  SUPPORT_SHARE
                    CALLF   mqerCI4
                    MOV     WORD PTR SUPPORT_VOTES,AX
                    MOV     WORD PTR SUPPORT_VOTES+2,DX
```

5.7.14. Programm zur Abspaltung des gebrochenen Teils mqerICX

Eingangsparameter

Die zu bearbeitende Gleitkommazahl steht auf der aktuellen Position des Register-Stack.

Funktion

Nicht normalisierte Werte werden vor der Operation normalisiert. Ist x bereits ein Integer-Wert, dann bleibt er unverändert. Ansonsten wird der gebrochene Teil abgetrennt. Auf diese Weise wird der Gleitkommazahl der betragsmäßig kleinere Integer-Wert zugeordnet.
Infinities bleiben unverändert. Es erfolgt keine Exception-Anzeige.

Ausgangsparameter

Die beschnittene Gleitkommazahl steht auf der aktuellen Position des Register-Stack.

Fehler

NaNs bewirken das Setzen des I-Bit im WM87-Statuswort. Wenn eine Beschneidung der Real-Zahl erfolgte, wird das P-BIT gesetzt.
Der Aufruf des Exception-Handlers erfolgt, wenn das entsprechende Exception-Bit unmaskiert ist. Das Operationscoderegister enthält dann 166H.

Programmbeispiel

```
                .
                :
                EXTRN   mqerICX:FAR
                .
                :
                DSEG
PRICF_MARK      RS      8               ; z.B. 101.489 Mark
ONE_HUNDRED     DD      0               ; 100.0
                .
                :
                CSEG
                .
                :
; Aus einem Mark-Betrag mit drei Stellen nach dem
; Komma wird ein Betrag, bestehend aus Mark und
; Pfennig durch Abtrennen der dritten Stelle nach
; dem Komma gebildet.
; Es wird vorausgesetzt, daß PRICE_MARK bereits
; initialisiert wurde.

                FLD64   PRICE_MARK
                FMUL32  ONE_HUNDRED
                CALLF   mqerICX
                FDIV32  ONE_HUNDRED
                FST64P  PRICE_MARK
```

5.7.15. Programm zum Runden real in word-integer mqerIE2

Eingangsparameter

Die zu rundende Gleitkommazahl steht auf der aktuellen Register-Stack-Position.

Funktion

Zunächst werden unnormalisierte Zahlen normalisiert. Daran erfolgt die Konvertierung der Gleitkommazahl in eine Integer-Zahl durch Runden. Liegt der Real-Wert genau zwischen zwei Integer-Zahlen (z. B. 4.5.), dann wird die geradzahlige Integer-Zahl genutzt. Negative Zahlen werden im Zweierkomplement dargestellt.

Beispiel

Gleitkommazahl	Integer-Zahl
3.1	3
10.5	10
-6.5	6

Fehler

Der Wertebereich der durch mqerIE2 konvertierbaren Zahlen erstreckt sich von -32768.5 bis kleiner 32767.5. Werte außerhalb dieses Bereiches, Infinities und NaNs bewirken das Setzen des I-Bit. Wurde das I-Exception maskiert, dann ist das Integer-

Indefiniti 8000H das Resultat.
Bezüglich des P-Exceptions gelten die bei mqerIA2 gegebenen Hinweise.
Bei Auftreten eines unmaskierten Exceptions erfolgt der Aufruf des Exception-Handlers. Im Operationscoderegister des WM87 steht danach 180H.

Programmbeispiel

```
                .
                :
                EXTRN   mqerIE2:FAR
                .
                :
                DSEG
INTEGER_VAR     DW      0
REAL_VAR        RS      8
                .
                :
                CSEG
                .
                :
; Es wird vorausgesetzt, daß REAL_VAR bereits
; mit einem Wert initialisiert wurde.
                .
                :
                FLD64   REAL_VAR
                CALLF   mqerIE2
                MOV     INTEGER_VAR,AX
```

5.7.16. Programm zum Runden real im short-integer mqerIE4

Eingangsparameter

Die zu rundende Gleitkommazahl steht auf der aktuellen Register-Stack-Position.

Funktion

Es gelten die Ausführungen von mqerIE2 mit dem Unterschied, daß bei mqerIE4 eine Integer-Zahl der Datenart short-integer erzeugt wird.

Ausgangsparameter

Die Integer-Zahl steht in den Registern DX und AX. Das höchstwertige Byte der Zahl befindet sich im DH-, das niederwertigste im AL-Register.

Fehler

Die zu konvertierende Real-Zahl muß der Bedingung

$$-2147483648.5 \leq x < 2147483647.5$$

entsprechen. Werte außerhalb dieses Bereiches, Infinities und NaNs verursachen das I-Exception. Wurde das I-Bit im Steuerwort maskiert, erzeugt der Prozessor das Indefinity 8 000 000H.
Bei Aufruf des Exception-Handlers (unmaskierte Exceptions) enthält das Operationscoderegister des Prozessors 17BH.

Programmbeispiele

```
                  .
                  :
                  EXTRN    mqerIE4:FAR
                  .
                  :
                  DSEG
                  .
                  :
COUNT             DD       0
REAL_COUNT        RS       8
                  .
                  :
                  CSEG
                  .
                  :
; Es wird vorausgesetzt, daß REAL_COUNT bereits
; einen gültigen Wert enthält.
                  .
                  :
                  FLD64    REAL_COUNT
                  CALLF    mqerIE4
                  MOV      WORD PTR COUNT,AX
                  MOV      WORD PTR (COUNT+2),DX
```

5.7.17. Programm zum Runden real in integer mqerIEX

Eingangsparameter

Die zu rundende Real-Zahl steht auf der aktuellen Register-Stack-Position.

Funktion

Es gelten die unter mqerIAX getroffenen Aussagen mit der Einschränkung, daß Werte, die genau zwischen zwei Integer-Werten liegen, auf den geradzahligen Integer-Wert gerundet werden.

Ausgangsparameter

Die gerundete Gleitkommazahl steht auf der aktuellen Register-Stack-Position.

Fehler

Bei erfolgter Rundung und bei Auftreten von NaNs werden die entsprechenden Bits im Statuswort des Prozessors gesetzt. Beide Werte bleiben unverändert.
Der Fehlercode im Operationscoderegister bei Aufruf des Exception-Handlers (unmaskiertes Exception) ist auf 178H gesetzt.

Programmbeispiel

```
                .
                :
                EXTRN   mqerIEX:FAR
                .
                :
                DSEG
                .
                :
THOUSANDS       RS      8               ;
UNITS           RS      8               ; Testwert
ONE_GRAND       DD      0               ; 1000.0
                .
                :
                CSEG
                .
                :
; Vom Testwert sind die Potenzen 10^1, 10^2 und 10^3
; abzuspalten. Es wird vorausgesetzt, daß UNITS
; bereits initialisiert wurde.
                FLD64   UNITS
                FDIV32  ONE_GRAD        ; Division durch 1000
                CALLF   mqerIEX
                FST64P  THOUSANDS
```

5.7.18. Programm zur Berechnung des Dezimalexponenten mqerLGD

Eingangsparameter

Die Gleitkommazahl steht auf der aktuellen Register-Stack-Position.

Funktion

Das Programm berechnet den Dezimalexponenten einer positiven Gleitkommazahl. Im Affin-Modus (vorzeichenbehaftete Infinities) ist +Infinity zulässig. Der Dezimalexponent ist ebenfalls +Infinity.

Ausgangsparameter

Der Dezimalexponent steht als Real-Zahl auf der aktuellen Register-Stack-Position.

Fehler

Folgende Werte bewirken ein invalid operation (I-Bit):

- negative Werte (einschließlich -Infinity)
- NaNs
- Unnormales
- ±Infinity (im Projektiv-Modus)

Das Ergebnis ist bei maskierten I-Exception NaN, wenn der Eingangswert ein NaN war und Indefinite bei allen anderen Werten. Das Z-Exception (Zerodivide) wird angezeigt, wenn der Eingangswert ± Null ist. In maskierten Fall ist -Infinity das Ergebnis.

Wenn der Eingangsparameter ein Denormal ist, testet das Programm das D-Bit im Steuerwort. Ist das D-Exception unmaskiert (Normalisierungsmodus), wird der Eingangswert als unnormal angesehen. Das D-Bit im Statuswort wird durch mqerLGD nicht gesetzt.
Das Operationscoderegister enthält bei Aufruf des Exception-Handlers (I-, Z-Exceptions unmaskiert) den Fehlercode 16DH.

Programmbeispiel

```
                .
                :
                EXTRN   mqerLGD:FAR
                .
                :
                DSEG
                .
                :
QUANTITY        RS      8               ; Potenz
TENS_POWER      RS      8               ; Exponent
                .
                :
                CSEG
                .
                :
; Es wird vorausgesetzt, daß QUANTITY die
; Potenz enthält
;
                FLD64   QUANTITY
                CALLF   mqerLGD
                FST64P  TENS_POWER
```

5.7.19. Programm zur Berechnung des Logarithmus naturalis mqerLGE

Eingangsparameter

Die Potenz steht als Gleitkommazahl auf der aktuellen Register-Stack-Position.

Funktion

Das Programm berechnet den Exponenten der Potenz zur Basis e. Die Konstante e hat den Wert +2.718281828459. Der Logarithmus wird als natürlich bezeichnet, weil er sich bei Berechnungen wie der inverse Differenzialquotient von 1/x verhält. Der Exponent ist für positive Potenzen definiert. Im Affin-Modus (vorzeichenbehaftete Infinities) liefert die Potenz +Infinity den Exponenten +Infinity.

Ausgangsparameter

Der Logarithmus steht als Gleitkommazahl auf der aktuellen Register-Stack-Position.

Fehler

Bei Aufruf des Exception-Handlers enthält das Operationscoderegister den Fehlercode 16CH. Alle übrigen Fehlerreaktionen entsprechen denen von mqerLGD.

Programmbeispiel

```
                 .
                 :
                 EXTRN  mqerLGE:FAR
                 .
                 :
                 DSEG
                 .
                 :
X                RS     8            ; Testwert
THETA            RS     8
ONE              DD     0            ; 1.0
                 .
                 :
                 CSEG
;
; Im folgenden wird der inverse hyperbolische Sinus
; von x berechnet. X wurde bereits mit einem
; entsprechenden Wert initialisiert.
;
                 .
                 :
                 FLD64  X
                 FMUL64 ST,ST
                 FADD32 ONE
                 FSQRT
                 FADD64 X             ; x + SQRT (x ** 2 + 1)
                 CALLF  mqerLGE
                 FST64P THETA
```

5.7.20. Programm zur Berechnung des Maximums mqerMAX

Eingangsparameter

Die beiden Werte stehen auf der aktuellen Stack-Position und der aktuellen Stack-Position + 1 des Register-Stack.

Funktion

mqerMAX liefert die größere der beiden Zahlen. Wenn x und y die gleichen Werte aber unterschiedliche Formate haben, wie z. B. +0 und -0, ist x das Ergebnis.
Wenn der Prozessor im Affin-Modus arbeitet, können ein oder beide Eingangswerte Infinities sein. Im Projektiv-Modus müssen x und y Infinity sein.

Ausgangsparameter

Die aktuelle Register-Stack-Position wird um 1 erhöht. Sie enthält den größeren Wert der beiden Eingangsparameter.

Fehler

Unnormales und Denormales führen zum D-Exception. Arbeitet der Prozessor im Warning-Modus (D maskiert), wird in mqerMAX fortgefahren. Das D-Bit im Statuswort bleibt gesetzt.

Beim Normalisierungsmodus (D unmaskiert) wird der Exception-Handler aufgerufen. Der Aufruf erfolgt direkt von dem Befehl, der zum D-Exception führte (FCOM64). Das Operationscoderegister des WM87 enthält den Fehlercode von FCOM64. Die Eingangswerte stehen noch im Register-Stack. Der Exception-Handler sollte die Normalisierung durchführen, den Befehl wiederholen und in mqerMAX fortfahren. Die Programme der EH87.LIB (siehe Abschnitt 6.), entsprechen diesen Forderungen.

Eingangsseitige NaNs führen zu invalid operation. Wurde das Ereignis maskiert, ist das NaN das Ergebnis. Sind beide Eingangswerte NaN, dann ist es das größere.
Das I-Bit wird auch gesetzt, wenn der Prozessor im Projektiv-Modus arbeitet und die Eingangswerte ein Infinity sind. mqerMAX liefert, wenn das Exception maskiert war, Indefinite als Ergebnis.

Bei Aufruf des Exception-Handlers, der durch ein I-Exception hervorgerufen wurde, stehen die Eingangsparameter noch im Register-Stack. Der Fehlercode im Operationscoderegister ist in jedem Falle 282H.

Programmbeispiel

```
                .
                :
                EXTRN   mqerMAX:FAR
                .
                :
                DSEG
                .
                :
VAR1            RS      8
VAR2            RS      8
LARGEST         RS      8
                .
                :
                CSEG
                .
                :
;
; mqerMAX berechnet das Maximum von VAR1 und VAR2.
; Es wird vorausgesetzt, daß VAR1 und VAR2 bereits
; entsprechende Werte enthalten.
;
                .
                :
                FLD64   VAR1
                FLD64   VAR2
                CALLF   mqerMAX
                FST64P  LARGEST
                .
                :
```

5.7.21. Programm zur Berechnung des Minimums mqerMIN

Eingangsparameter

Die beiden Eingangsparameter stehen auf der aktuellen Register-Stack-Position bzw. auf der aktuellen Register-Stack-Position + 1.

Funktion

mqerMIN liefert den kleineren der beiden Eingangswerte. Wenn x und y die gleichen Werte, aber unterschiedliche Formate haben, wie z. B. +0 und -0, ist x das Ergebnis.
Im Affin-Modus können ein oder beide Eingangswerte Infinity sein. Im Projektiv-Modus müssen x und y Infinity sein.

Ausgangsparameter

Die aktuelle Register-Stack-Position wird um 1 erhöht. Sie enthält das Minimum der beiden Eingangswerte.

Fehler

Bezüglich der Exceptions gelten die bei mqeMAX getroffenen Ausführungen. Der Fehlercode ist 281H.

5.7.22. Programm zur Berechnung des Restes einer Division mqerMOD

Eingangsparameter

Dividend und Divisor stehen im Register-Stack auf der aktuellen Position bzw. der aktuellen Position + 1.

Funktion

mqerMOD berechnet den Wert des Ausdruckes

y - (x * mqerICX(y / x)).

Es handelt sich dabei um den verbleibenden Rest der Division y / x. Das Ergebnis weist keinen Rundungsfehler auf. Das Vorzeichen von x ist bedeutungslos.
Die Berechnung erfolgt durch die Subtraktion y - nx. Der Faktor n ergibt sich aus:

n = mqerICX(y / x).

Das Vorzeichen des Ergebnisses entspricht den Bedingungen:

$y \geq 0$: $0 \leq$ Ergebnis $< x$
$y < 0$: $x <$ Ergebnis ≤ 0

Beispiele

x	y	Ergebnis
±5	-10	0
±5	-19.99	-4.99
±5	+44.75	+4.75

Der x-Wert darf ein Infinity sein. Das Ergebnis ist dann ab/hangig vom y-Wert und dem D-Bit im Steuerwort. Bei normalisiertem und endlichen y ist y das Ergebnis. Bei unnormalisiertem y ist das normalisierte y das Resultat. Bei denormalisiertem y und unmaskiertem D-Exceptin ist das Ergebnis 0. Ist D maskiert, bildet das denormalisierte y das Resultat. In keinem Fall wird vpn mqerMOD ein Fehler angezeigt.

Ausgangsparameter

Das Ergebnis steht auf der um 1 erhöhten Register-Stack-Position.

Fehler

NaNs führen zum I-Exception. Bei maskiertem I-Bit liefert mqerMOD das NaN als Ergebnis. Sind beide Eingangswerte NaNs, wird das größere genutzt.
Invalid operation tritt auch auf, wenn x ein Unnormal, ein Denormal, Null oder wenn y ein Indefnite ist. Wenn das I-Exception maskiert ist, liefert mqerMOD Indefinite.
In all diesen Fällen stehen bei Aufruf des Exception-Handlers (I-Exception unmaskiert) die Eingangswerte unverändert im Register-Stack. Der Fehlercode im Operationscoderegister ist 269H.
Denormalisierte y-Werte und unnormalisierte y-Werte, die durch Normalisierung zu Denormals führen, bewirken daß U-Exception. Das Ergebnis bei aufgetretenem maskierten U-Exception ist 0. Unmaskierte U-Exceptions führen zum Aufruf des Exception-Handlers. Der Fehlercode ist 169H. Die aktuelle Register-Stack-Position enthält das modifizierte Ergebnis, von dessen Exponent 24576 zu subtrahieren sind, um das wahre Ergebnis zu erhalten.

Programmbeispiel

```
                .
                :
                EXTERN mqerMOD:FAR
                .
                :
                DSEG
LAST_THREE      RS      8
Y               RS      8               ; Testwert y
ONE_GRAND       RS      8               ; x
                .
                :
                CSEG

; Im folgenden wird y Modulo x berechnet.
; Es wird vorausgeetzt, daß LAST_THREE und ONE_GRAND
; bereits entsprechende Werte enthalten.

                FLD64   Y
                FLD64   ONE_GRAND
                CALLF   mqerMOD
                FST64P  LAST_THREE
```

5.7.23. Programm zur Berechnung des Restes einer Division mqerRMD

Eingangsparameter

Dividend und Divisor stehen im Register-Stack auf der aktuellen Position bzw. der aktuellen Position + 1.

Funktion

mqerRMD berechnet den Wert des Ausdruckes

y - (x * mqerIEX(y / x)).

Er stellt den Rest der Division y / x dar. Der Quotient wurde durch mqerIEX auf einen ganzzahligen Wert gerundet. Das Ergebnis selbst weist keinen Rundunsfehler auf. Das Vorzeichen von x ist ohne Bedeutung.

Beispiele

x	y	Ergebnis
±5	-7	-2
±5	-10	0
±5	-19.99	+0.01
±5	2	2
±5	4	-1
±5	44.75	-.25

Wenn y ein ungerades ganzzahliges Vielfaches von x/2 ist, kann das Ergebnis +/-x/2 sein. Der Wert wird durch mqerIEX festgelegt, das bei gleichen Abständen auf geradzahlige Integer-Werte rundet.

y-Werte in der Form (... -7x/2, -3x/2, x/2, 5x/2, 9x/2 ...) führen zu x/2. y-Werte in der Form (... -5x/2, -x/2, 3x/2, 7x/2, 11x/2 ...) haben -x/2 zum Ergebnis.

Die x-Werte können Infinities sein. In diesen Fällen liefert mqerRMD y als Ergebnis, wenn y endlich und normalisiert ist. Ist y ein Unnormal, erfolgt die Normalisierung. Bei denormalisierten y-Werten ist das Ergebnis vom D-Bit im Steuerwort abhängig. Im Normalisierungmodus (D unmaskiert) ist das Resultat Null. Im Warning-Modus (D maskiert) bildet der denormalisierte y-Wert das Ergebnis. In beiden Fällen erfolgt keine Fehlermitteilung.

Ausgangsparameter

Das Ergebnis steht auf der um 1 erhöhten aktuellen Register-Stack-Position.

Fehler

Eingangsseitige NaNs führen zum I-Exception. mqerRMD liefert bei maskiertem I-Bit das NaN als Ergebnis. Sind beide Eingangswerte NaNs, wird das größere genutzt.
Das I-Exception tritt auch auf, wenn x ein Denormal, ein Unnormal, Null oder wenn y ein Infnity ist. Im maskierten Fall liefert mqerRMD Indefinite als Ergebnis.

Alle unmaskierten I-Exceptions führen zum Aufruf des Exception-Handlers. Die Eingangswerte stehen noch im Register-Stack. Der Fehlercode im Oprationscoderegister ist 27AH.
Denormalisierte y-Werte und unnormalisierte y-Werte, die durch Normalisierung zu Denormals führen, bewirken das U-Exception.
Das Ergebnis bei aufgerundetem maskierten U-Exception ist Null. Unmaskierte U-Exceptions führen zum Aufruf des Exception-Handlers. Der Fehlercode ist 17AH. Die aktuelle Register-Stack-Position enthält das modifizierte Ergebnis, mit einem um 24576 vergrößerten Exponenten.

Programmbeispiel

```
                .
                :
                EXTERN mqerRMD:FAR
                .
                :
                DSEG
                .
                :
THETAA          RS      8
                .
                :
                CSEG
                .
                :
; Das folgende Beispiel dient zur Reduzierung eines
; Winkels im Bogenmaß auf den Bereich zwischen +Pi und
; -Pi. Es wird vorausgesetzt, das THETA bereits einen
; entsprechenden Wert (z. B. -6.0) enthält.

                FLD64  THETA          ; z. B. -6.0
                FLDPI                 ; Pi
                FADD64 ST,ST          ; 2PI
                CALLF  mqerRMD
                FST64P THETA          ; +0.2831853
                .
                :
```

5.7.24. Programm zur Bildung von y mit dem Vorzeichen von x mqerSGN

Eingangsparameter

Die Eingangswerte x und y stehen im Register-Stack auf der aktuellen Position bzw. der aktuellen Position + 1.

Funktion

Für x größer oder gleich Null ist der Betrag von y das Ergebnis. Falls x kleiner als Null ist, liefert mqerSGN den Betrag von y mit negativen Vorzeichen.
Die x-Werte +0, -0 und Pseudonullen führen zu positiven Ergebnissen.
Unnormalisierte x-Werte, ausgenommen Pseudonullen, sind zulässig. Das Ergebnis ist dann der Betrag von y mit dem Vorzeichen von x. Ist der x-Wert ein Infinity, so bestimmt er das Vorzeichen des

Ergebnisses sowohl im Affin- als auch im Projektiv-Modus.
y kann alle Werte annehmen. Sie bleiben bis auf das Vorzeichen unverändert.

Ausgangsparameter

Das Ergebnis steht auf der um 1 erhöhten aktuellen Register-Stack-Position.

Fehler

Der FTST-Befehl innerhalb von mqerSGN bewirkt ein D-Exception, wenn der x-Wert ein Denormal ist. Im Warning-Modus (D maskiert) erhält das Resultat das Vorzeichen von x. Der Normalisierungsmodus (D unmaskiert) bewirkt den Aufruf des Exception-Handlers. Die Eingangswerte stehen noch im Register-Stack. Der Exception-Handler sollte den x-Wert normalisieren, den FTST-Befehl wiederholen und in mqerSGN fortsetzen.
Die Routinen EH87.LIB ersetzen Denormals durch 0. Das Ergebins von mqerSGN ist demzufolge in jedem Falle positiv.
NaNs als x-Werte führen zum I-Exception. Bei maskiertem I-Bit im Steuerwort bleibt das NaN erhalten. Bei unmaskiertem I-Bit wird der Exception-Handler aufgerufen. Der Fehlercode ist 264H. Die Eingangswerte stehen noch im Register-Stack.

Programmbeispiel

```
                 .
                 :
                 EXTERN mqerSGN:FAR
                 .
                 :
                 DSEG
                 .
                 :
THETA            RS      8
X_COOR           RS      8               ; Testwerte
                 .
                 :
                 CSEG
                 .
                 :
; Die folgenden Operationen liefern Pi mit dem Vorzeichen
; von X_COOR als Ergebnis. Es wird vorausgesetzt, daß
; X_COOR bereits einen entsprechenden Wert enthält.
                 .
                 :
                 FLDPI                   ; PI
                 FLDL64 X_COOR
                 CALLF  mqerSGN
                 FST64P THETA
```

5.7.25. Programm zur Berechnung des Sinus mqerSIN

Eingangsparameter

Die aktuelle Register-Stack-Position enthält den Winkel im Bogenmaß.

Funktion

mqerSIN berechnet den Sinus von x. Der Winkel ist im Bogenmaß anzugeben. Nullen, Pseudonullen, Denormals und Unnormals, deren Wert kleiner als 2^{-63} ist, als Eingangswerte bilden unverändert das Ergebnis.

Ausgangsparameter

Der Sinus des Winkels steht auf der aktuellen Register-Stack-Position.

Fehler

Infinities, NaNs und Unnormals, deren Wert nicht kleiner als 2^{-63} ist, erzeugen ein I-Exception. Wurde das Exception maskiert, führen NaNs zu Nans und alle anderen Eingangswerte zu Indefinies als Ergebnis. Bei rückgesetztem I-Bit im Steuerwort erfogt der Aufruf des Exception-Handlers. Der Fehlercode im Operationscoderegister ist 171H.

Programmbeispiel

```
                .
                :
                EXTERN     mqerSIN:FAR
                .
                :
                DSEG
                .
                :
POLAR_THETA     RS         8              ; Polarkoordinaten, Winkel in
;                                           Grad
POLAR_R         RS         8              ; Polarkoordinaten-Radius
REC_Y           RS         8              ; y-Koordinate
DEC_TO_RAD      RS         8              ; PI/180
                .
                :
                CSEG
                .
                :
; Im folgenden wird die y-Koordinate eines x-y-Koordinaten
; systems aus den gegebenen Polarkoordinaten berechnet.

                FLD64      POLAR_THETA    ; Winkel in Grad
                FLD64      DEG_TO_RAD     ;
                FMUL
                CALLF      mqerSIN        ; Bogenmaß
                FMUL64P    POLAR_R        ; y-Koordinate
                FST64P     REC_Y
```

5.7.26. Programm zur Berechnung des Sinus Hyperbolicus mqerSNH

Eingangsparameter

Die aktuelle Register-Stack-Position enthält den Winkel im Bogenmaß.

Funktion

mqerSNH liefert den hyperpolischen Sinus vom Winkel x im Bogenmaß. Eingangsseitige Nullen, Pseudonullen, Infinities, Denormals und Unnormals, die kleiner als 2^{-63} sind, bilden unverändert das Ergebnis.

Augangsparameter

Der Wert des hyperbolischen Sinus steht auf der aktuellen Register-Stack-Position.

Fehler

NaNs und Unnormals, die nicht kleiner als 2^{-63} sind, erzeugen das I-Exception. Wurde das Exception maskiert, bleiben NaNs erhalten, Unnormals führen zu Indefnities.
Eingangswerte, die größer als etwa 11355 oder kleiner als -11355 sind, bewirken das O-Exception (Überlauf). Bei maskiertem O-Bit haben positive Eingangswerte +Infinity und negative -Infinity zum Ergebnis. Bei maskiertem O-Exception wird der Exception-Handler aufgerufen. Der Fehlercode ist 16EH. Die Eingangspaameter stehen noch auf dem Register-Stack.

Programmbeispiel

```
                .
                :
                EXTERN mqerSHN:FAR
                .
                :
                DSEG
                .
                :
INPUT_VALUE     RS     8
OUTPUT_VALUE    RS     8
                .
                :
                CSEG
                .
                :
; Im folgenden wird der hyperbolische Sinus von INPUT_VALUE
; berechnet.
                .
                :
                FLD64  INPUT_VALUE
                CALLF  mqerSNH
                FST64P OUTPUT_VALUE
```

5.7.27. Programm zur Berechnung des Tangens mqerTAN

Eingangsparameter

Die aktuelle Register-Stack-Position enthält den Winkel im Bogenmaß.

Funktion

mqerTAN berechnet den Tangens von x. Der Winkel x ist im Bogenmaß anzugeben. Nullen, Pseudonullen, Denormals und Unnormals, die kleiner als 2^{-63} sind, sind mit dem Tangens identisch.

Ausgangsparameter

Der Tangens steht auf der aktuellen Register-Stack-Position.

Fehler

Infinities, NaNs und Unnormals, dessen Werte größer oder gleich 2^{-63} sind, erzeugen ein I-Exception. Wurde das Exception maskiert, bleiben NaNs erhalten, die anderen Werte führen zu Indefinite.
Unnormalisierte I-Exceptions bewirken den Aufruf des Exception-Handlers. Der Eingangswert ist im Register-Stack. Das Operationscoderegister des WM87 enthält den Fehlercode 173H.
Eingangswerte, die ein ungerades Vielfaches von PI/2 (dargestellt in temporary-real) sind, verursachen ein Z-Exception. Wurde Z maskiert, liefert mqerTAN +Infinity. Im anderen Fall wird der Exception-Handler aufgerufen. Der Eingangswert steht im Register-Stack. Der Fehlercode ist 173H.

Programmbeispiel

```
                .
                :
                EXTERN mqerTAN
                .
                :
                DSEG
THETA_DEG       RS     8
SLOPE           RS     8
DEG_TO_RAD      RS     8               ; PI/180
                .
                :
                CSEG
                .
                :
; Im folgenden wird der Tangens von THETA_DEG berechnet.
; Es wird vorausgesetzt, daß THETA_DEG und DEG_TO_RAD
; bereits entsprechende Werte enthalten.

                FLD64  THETA_DEG       ; Winkel in Grad
                FLD64  DEG_TO_RAD
                FMUL
                CALLF  mqerTAN         ; Winkel im Bogenmaß
                FST64P SLOPE
                .
                :
```

5.7.28. Programm zur Berechnung des Tangens Hyperbolicus mqerTNH

Eingangsparameter

Die aktuelle Register-Stack-Position enthält den Winkel x.

Funktion

mqerTNH berechnet den hyperbolischen Tangens. Der Winkel x ist im Bogenmaß anzugeben. Die x-Werte Null, Pseudonull, Denormal und Unnormals, die kleiner als 2^{-63} sind, stellen zugleich das Ergebnis dar. Infinities sind als Eingangswerte zulässig. Sie ergeben 1 mit dem Vorzeichen des Infinities. Das gilt auch im Projektiv-Modus.

Ausgangsparameter

Das Ergebnis steht auf der aktuellen Register-Stack-Position.

Fehler

NaNs und Unnormals, die nicht kleiner als 2^{-63} sind, bewirken das I-Exception. Wurde das Exception maskiert, bleiben NaNs erhalten; unzulässige Unnormals werden in Indefinite überführt. Bei rückgesetztem I-Bit im Steuerwort erfolgt der Aufruf des Exception-Handlers mit dem Fehlercode 170H. Der Register-Stack enthält den Eingangswert.

Programmbeispiel

```
                .
                :
                EXTRN  mqerTNH.FAR
                .
                :
                DSEG
                .
                :
INPUT_VALUE     RS     8
OUTPUT_VALUE    RS     8
                .
                :
                CSEG
                :
                .
                FLD64  INPUT_VALUE
                CALLF  mqerTNH
                FST64P OUTPUT_VALUE
                .
                :
```

5.7.29. Programm zur Berechnung einer Potenz mqerY2X

Eingangsparameter

Die Werte von Exponent und Basis stehen im Register-Stack auf der aktuellen Position bzw. der aktuellen Position + 1.

Funktion

mqerY2X berechnet die Potenz y^x im Datenformat real. Basis der Berechnung ist die Formel $2^{x*LG2(y)}$. Daraus ergibt sich, daß y nur Werte größer als Null annehmen darf. Die folgenden drei Fälle stellen Ausnahmen dar:

- Bei x größer als Null und y gleich Null ist das Ergebnis Null.

- Bei x gleich Null kann y negative Werte annehmen. Das Ergebnis ist immer 1.

- Wenn x eine Integer-Zahl ist, kann y negativ sein. Die Berechnung erfolgt gemäß der obigen Formel mit dem Betrag von y. Das Ergebnis ist positiv, wenn x geradzahlig ist. Bei ungeradzahligem x-Wert ist das Ergebnis negativ.

Nullen, Infinities, Unnormals und Denormals sind unter bestimmten Voraussetzungen als Eingangswerte zulässig.
Ist einer der Eingangswerte denormalisiert, wird er in Abhängigkeit vom D-Bit im Steuerwort durch Null (D unnormalisiert) oder das entsprechende Unnormal (D maskiert) ersetzt. Die Anzeige eines D-Exceptions im Statuswort des WM87 erfolgt nicht.
Ein unnormalisierter y-Wert ist nur erlaubt, wenn x entweder eine normalisierte Integer-Zahl oder ein Infinity ist. Wenn x ein Infinity ist, wird die Funktion berechnet, als wäre y der entsprechende normalisierte Wert.
Wenn x Null ist, muß y ungleich Null sein. Das Ergebnis ist 1.
Ist x eine Integer-Zahl ungleich Null mit einer Länge von maximal 32 Bit, wird der Funktionswert durch mqerYI2 (s. u.) berechnet.
Ein unnormalisierter x-Wert ist nur zulässig, wenn er ungleich Null und wenn y Null oder Infinity ist. Die Berechnung wird dann mit dem normalisierten x-Wert durchgeführt.
Im Affin-Modus gibt es eine Anzahl von Fällen, in denen Infinities als Eingangswerte erlaubt sind:

- Wenn y -Infinity ist, muß x eine Integer-Zahl ungleich Null sein. Für positive x-Werte ist der Betrag des Ergebnisses Infinity, für negative Null. Das Vorzeichen ist bei geradzahligem x positiv, bei ungeradzahligem negativ.

- Wenn y +Infinity ist, sind alle x-Werte, die ungleich Null sind, erlaubt. Für positive x ist das Ergebnis +Infinity. Negative x-Werte ergeben Null.

- Wenn x +Infinity ist, muß y positiv oder Null, aber ungleich 1 sein. Das Ergebnis ist Null bei y-Werten, die kleiner als 1 sind. Größere y-Werte als 1 ergeben +Infinity.

- Wenn x -Infinity ist, muß y ebenfalls positiv oder Null und ungleich 1 sein. y-Werte, die kleiner 1 sind, ergeben Infinity. Werte größer als 1 liefern -Infinity als Ergebnis.

Im Projektiv-Modus sind nur in einem Falle Infinities als Eingangswerte erlaubt. Wenn y ein Infinity ist, muß x eine Integer-Zahl sein, die ungleich Null ist. Das Ergebnis entspricht dem im Affin-Modus.

Ausgangsparameter

Das Ergebnis steht auf der um 1 erhöhten Register-Stack-Position.

Fehler

Eingangsseitige NaNs erzeugen ein I-Exception. Wurde das Exception maskiert, liefert mqerY2X das NaN als Ergebnis. Waren beide Eingangswerte NaNs, wird der größere Wert genutzt.
Unnormals, Infinities und negative y-Werte sind als Eingangsparameter möglich (s. o.). In den Fällen, in denen sie nicht zulässig sind, führen sie zum I-Exception und ergeben bei maskierten I-Bit Indefinite als Ergebnnis. Dazu gehören z. B. die Wertekombinationen:

y = 1, x = Infinity
y = 0, x = 0

Die Eingangswerte y = 0, x < 0 (einschließlich-Infinity) bewirken das Z-Exception mit +Infinity als Ergebnis bei maskiertem Z-Bit. Wenn y^x im Format temporary-real nicht dargestellt werden kann, wird das O-bzw. U-Exception angezeigt. Ein Overflow führt bei maskiertem O-Bit zu +Infinity. Der Underflow hat entweder ein Denormal oder Null zum Ergebnis. Die Exceptions I, O, U und Z bewirken den Aufruf des Exception-Handlers, wenn die entsprechenden Maskenbits im Steuerwort rückgesetzt sind. Der Fehlercode im Operationscoderegister ist immer 26AH. Die Eingangswerte verbleiben im Register-Stack.

Programmbeispiel

```
                .
                :
                 EXTRN mqerY2X:FAR
                .
                :
                DSEG
                .
                :
INPUT_VALUE     RS      8               ; Testwert
CUBE_ROOT       RS      8
ONE_TMIRD       RS      8               ; 0.333 ... 3
                .
                :
                CSEG
                .
                :
; In folgenden wird die dritte Wurzel von INPUT_VALUE
; berechnet. Die entsprechenden Werte stehen bereits
; im INPUT_VALUE und ONE_TMIRD
                FLD64  INPUT_VALUE
                FLD64  ONE_TMIRD
                CALLF  mqerY2X
                FST64P CUBE_ROOT
                .
                :
```

5.7.30. Programm zur Berechnung einer Potenz mqerYI2

Eingangsparameter

Der Basiswert steht auf der aktuellen Register-Stack-Position. Das Register AX enthält den Exponenten.

Funktion

mqerYI2 berechnet die Potenz mit der Basis in der Datenart temporary-real und dem Exponenten in der Datenart word-integer. Vor der Brechnung erfolgt eine Prüfung auf spezielle Werte.
Der Exponent Null führt in jedem Falle zum Ergebnis 1. Ist die Basis +Infinity, dann ist das Ergebnis +Infinity bei positiven und 0 bei negativen Exponenten. Ein Exception wird nicht angezeigt.
Ist die Basis -Infinity, dann ist der Betrag des Ergebnisses Infinity für positive und Null für negative Exponenten. Das Vorzeichen ist + bei geradzahligem und - bei ungeradzahligem Exponenten.
Die Basiswerte ±0 führen zu -0, wenn x - 0 und AX ungeradzahlig sind. In allen anderen Fällen ist das Ergebnis +0.
Denormalisierte x-Werte verursachen kein Exception. Es findet jedoch eine Prüfung des D-Bit im Steuerwort statt. Im Normalisierungsmodus (D unmaskiert) wird der x-Wert durch Null ersetzt. Im Warning-Modus (D maskiert) wird der Basiswert in das entsprechende Unnormal überführt. Die Berechnung erfolgt mit den neuen Werten.
Bei unnormalisierten x-Werten und negativem Exponenten wird zunächst 1/x unter Berücksichtigung der unnormalisierten Stellen berechnet. Danach erfolgt die Berechnung der Potenz durch Quadrierung und Multiplikation.
Bei unnormalisierten x-Werten und positivem Exponenten berechnet mqerYI2 die Potenz direkt durch Quadrierung und Multiplikation.
Für normalisierte x-Werte, die ungleich Null sind, erfolgt die Berechnung der Potenz auf unterschiedlichen Wegen. Ist der Exponent größer als 63 oder kleiner als -63, berechnet mqerYI2 das Ergebnis mit Hilfe des Logerithmus gemäß der Formel

$$2^{AX*LG2(x)}$$

Für Exponentenwerte zwischen 1 und 63 ergibt sich das Ergebnis durch mehrfache Quadrierung und Multiplikation.
Potenzen, deren Exponent zwischen -63 und -1 liegt, können entsprechend dem Ausdruck

$$\frac{1}{x * x * x \ldots x}$$

berechnet werden. mqerYI2 prüft zuvor, ob dabei Exceptions auftreten würden. Wäre das der Fall, erfolgt die Berechnung gemäß

$$\frac{1}{x} * \frac{1}{x} \ldots \frac{1}{x}$$

Treten dabei Exceptions auf, wird der Exception-Handler aufgerufen.
Die maximale Anzahl durchgeführter Multiplikationen bzw. Quadrierungen ist 9.

Ausgangsparameter

Der Wert der Potenz steht auf der aktuellen Register-Stack-Position.

Fehler

Ein I-Exception wird angezeigt, wenn x ein NaN ist. Bei maskiertem I-Bit bleibt das NaN im Register-Stack.
Ergebnisse, deren Betrag den kleinsten möglichen Wert im Temporary-Real-Format unterschreiten, verursachen ein U-Exception mit Denormals bzw. Null als Resultat.
Ein Overflow wird bei Überschreitung des Zahlenbereiches angezeigt. Der Prozessor reagiert mit + bzw. -Infinity, wenn das O-Bit maskiert wurde
Beim Auftreten von I-, O- bzw. U-Exceptions und unmaskierten Exception-Bits im Steuerwert wird der Exception-Handler aufgerufen. Der Fehlercode ist 27CH. Der Integer-Exponent wird in das Temporary-Real-Format konvertiert und auf die aktuelle Register-Stack-Position gebracht. Der originale x-Wert steht auf dem nächsten Platz.

Programmbeispiel

```
                  .
                  :
                  EXTRN   mqerYI2:FAR
                  .
                  :
                  DSEG
                  .
                  :
                  POWER  RS            8
                  REAL_BASE            RS8
                  REAL_OUTPUT          RS8.
                  .
                  :
                  CSEG
                  .
                  :
; Der Wert in REAL_BASE wird mit dem Inhalt von
; POWER potenziert
                  FLD64   REAL_BASE
                  MOV     AX,POWER
                  CALLF   mqerYI2
                  FST64P  REAL_OUTPUT
                  .
                  :
```

5.7.31. Programm zur Berechnung einer Potenz mqerYI4

Eingangsparameter

Der Wert der Basis der Potenz steht auf der aktuellen Register-Stack-Position. Die Register AX und DX enthalten den Integer-Exponenten. Das höherwertige Wort steht in DX.

Funktion

mqerYI4 berechnet die Potenz mit der Basis in der Datenart temporary-real und dem Exponenten in der Datenart short-integer. Ist der Exponent Null, dann ist das Ergebnis 1, unabhängig vom Wert der Basis.
Ist die Basis +Infinity, dann ist das Ergebnis +Infinity bei positivem und Null bei negativem Exponenten. Das Vorzeichen ist + bei geradzahligen und - bei ungeradzahligem Exponenten.
Die Basiswerte ±0 führen zu -0, wenn x - 0 und AX ungeradzahlig sind. In allen anderen Fällen ist das Ergebnis +0.
Denormalisierte x-Werte verursachen kein Exception. Es findet jedoch eine Prüfung des D-Bits im Steuerwort des WM87-Prozessors statt. Im Normalisierungsmodus (D unmaskiert) wird der x-Wert durch Null ersetzt. Im Warning-Modus (D maskiert) wird der Wert der Basis in das entsprechende Unnormal überführt. Die Berechnung erfolgt mit den neuen Werten.
Bei unnormalisierten x-Werten und negativem Exponenten wird zunächst 1/x unter Berücksichtigung der unnormalisierten Stellen berechnet. Danach erfolgt die Berechnung der Potenz durch Quadrierung und Multiplikation.
Bei unnormalisierten x-Werten und positiven Exponenten berechnet mqerYI4 die Potenz direkt durch Quadrierung und Multiplikation.
Für normalisierte x-Werte, die ungleich Null sind, erfolgt die Berechnung der Potenz auf unterschiedlichen Wegen. Ist der Exponent größer als 63 oder kleiner als -63, berechnet mqerYI4 das Ergebnis mit Hilfe des Logarithmus gemäß der Formel

$$2^{AX*LG2(x)}$$

Für Exponentenwerte zwischen 1 und 63 ergibt sich das Ergebnis durch mehrfache Quadrierung und Multiplikation.
Bei Potenzen, deren Exponenten zwischen -63 und -1 liegen, prüft mqerYI4 zunächst, ob bei der Berechnung gemäß der Form

$$\frac{1}{x * x * x \ldots x}$$

Exceptions auftreten würden. Wäre das der Fall, wird das Ergebnis auf der Grundlage von

$$\frac{1}{x} * \frac{1}{x} * \frac{1}{x} * \ldots * \frac{1}{x}$$

ermittelt. Hierbei auftretende Exceptions führen zum Aufruf des Exception-Handlers.
Die maximale Anzahl durchgeführter Multiplikationen ist 9.

Ausgangsparameter

Der Wert der Potenz steht auf der aktuellen Register-Stack-Position.

Fehler

Das I-Exception wird angezeigt, wenn x ein NaN ist. Bei maskiertem I-Bit bleibt das NaN im Register-Stack.
Ergebnisse, deren Beträge den kleinsten möglichen Wert in temporary-real unterschreiten, verursachen ein U-Exception mit Denormals bzw. Null als Resultat.
Ein Overflow wird bei Überschreitung des Zahlenbereichs angezeigt. Der Prozessor reagiert mit + bzw. -Infinity, wenn das

O-Bit maskiert wurde. Beim Auftreten von I-, U- oder O-Exceptions und unmaskierten Exception-Bits im Steuerwort wird der Exception-Handler aufgerufen. Der Fehlercode ist 27CH. Der Integer-Exponent wird in temporary-real konvertiert und auf die aktuelle Register-Stack-Position gebracht. Der originale x-Wert steht auf dem nächsten Platz.

Programmbeispiel

```
                .
                :
                EXTRN   mqerYI4:FAR
                .
                :
                DSEG
                .
                :
POWER           RS      8
REAL_BASE       RS      8
REAL_OUTPUT     RS      8
                .
                :
                CSEG
                .
                :
; Der Wert in REAL_BASE wird mit dem Inhalt von
; POWER multipliziert.
                .
                :
                FLD64   REAL_BASE
                MOV     AX, WORD PTR POWER
                MOV     DX, WORD PTR DOWER+2
                CALLF   mqerYI4
                FST64P  REAL_OUTPUT
                .
                :
```

5.7.32. Programm zur Berechnung einer Potenz mqerYIS

Eingangsparameter

Der Basiswert x steht auf der aktuellen Register-Stack-Position. Der Exponent ist vor dem Aufruf von mqerYIS als Integer-Zahl der Datenart word-integer auf den WM86-Stack zu bringen.

Funktion

Hier gilt das bei mqerYI2 gesagte.

Ausgangsparameter

Der Potenzwert steht im Register-Stack auf der aktuellen Position. Der Exponent ist aus dem WM86-Stack ausgetragen.

Fehler

siehe mqerYI2

Programmbeispiel

```
                :
                .
                EXTRN   mqerYIS:FAR
                .
                :
                DSEG
                .
                :
INTEREST_RATE   RS      8
NUMBER_OF_PERIODS       DW          0
START_AMOUNT    RS      8
FINISH_AMOUNT   RS      8
                .
                :
                CSEG
                .
                :
; Im folgenden wird ein Endbetrag, ausgehend von einem
; Anfangswert, einem Zinssatz und einer Anzahl von
; Zeitintervallen, berechnet.
                .
                :
                FLD1
                FADD64 INTERREST_RATE
                PUSH    NUMBER_OF_PERIODS
                .
                :
```

6. Exception-Handler-Bibliothek EH87

6.1. Vorbemerkungen

Die Bibliothek EH87 kann beim Schreiben von Exception-Handlern genutzt werden. Sie besteht aus den Teilprogrammen DECODE, NORMAL, ENCODE, SIEVE und FILTER, mit deren Hilfe sowohl Exception-Ursachen angezeigt als auch Fehler behoben werden können.

DECODE muß zu Beginn des Exception-Handlers aufgerufen werden. Es liefert den kompletten Status des WM87-Prozessors und die Operation, die das Exception verursachte, sowie deren Argumente bzw. Resultate.

NORMAL führt beim Auftreten eines D-Exceptions (denormalized operand) die Normalisierung durch. Damit kann im Hauptprogramm der Test auf nicht normalisierte Argumente entfallen.

SIEVE bietet Möglichkeiten zur Behandlung der beiden I-Exceptions "non-trapping-NaN" und "non-ordered comparison".

ENCODE wird am Ende des Exception-Handlers aufgerufen. Es stellt den durch DECODE geretteten Status des WM87-Prozessors wieder her und bereitet die Fortsetzung im trap-verursachenden Programm vor.

FILTER nutzt alle zuvor genannten Programme. Es ist dann vorteilhaft anwendbar, wenn durch den Exception-Handler nur die Leistungen von NORMAL und SIEVE genutzt, und weitere Fehler nur angezeigt werden sollen.

6.2. Begriffe

6.2.1. Non-trapping-NaNs

NaNs (not a number) sind Bitkombinationen, die nicht zur Datendarstellung genutzt werden. Sie existieren nur im Zahlenformat real (Gleitkomma) und sind gekennzeichnet durch maximalen Exponenten (alle Bits gesetzt) sowie einen von 1.0 ... 0B abweichenden Mantissenwert. Die große Anzahl von NaNs bietet die Möglichkeit, Diagnostikinformationen in diesen Werten zu verschlüsseln. Da NaNs Gleitkommaoperationen unverändert durchlaufen, ist es zu einem späteren Zeitpunkt möglich, die Ursache, die zum NaN führte, zu erkennen und geeignete Maßnahmen einzuleiten.
EH87 wertet das erste Bit der Mantisse des NaN als Steuerbit. Ist dieses Bit gesetzt, so wird das NaN als "non-trapping" betrachtet und im trap-verursachenden Programm fortgefahren. Auf diese Weise ist es möglich, mehrfache Exception-Meldungen, die ihre Ursache in ein und demselben NaN haben, zu unterdrücken.

Der WM87-Prozessor unterscheidet nicht zwischen "trapping" und "non-trapping" NaN. Alle NaN erzeugen ein I-Exception. Es ist Aufgabe des Exception-Handlers, entsprechend zu unterscheiden. Das Programm SIEVE der EH87 folgt diesen Festlegungen.

6.2.2. Non-ordered comparisons

Beim Vergleich zweier Real-Zahlen mittels der Befehle FCOM bzw. FTST wird ein I-Exception erzeugt, wenn es sich um NaN handelt. Um die entsprechende Fehlermeldung zu verhindern, werden folgende Maßnahmen vorgeschlagen:
Wenn dem Vergleichsbefehl ein MOV AX,AX folgt, dann muß der Exception-Handler "non-trapping" NaN als legale Operanden erlauben. Er muß weiterhin "non-ordered" (C3 = C0 = 1 im Statuswort) liefern, wenn beide Argumente die gleichen NaN sind, und das I-Exception ignorieren. Das Programm SIEVE entspricht diesen Festlegungen.

6.3. Arbeit mit den Programmen

6.3.1. Datenfeld ESTATE87

ESTATE87 ist ein 144-Byte-Feld, das durch DECODE aufgebaut und von allen Teilprogrammen der EH87 genutzt wird. Es enthält alle notwendigen Informationen für die Fehleruntersuchung durch den Exception-Handler, wie: Status des WM87-Prozessors, trapverursachende Operation, Werte und Formate der Argumente und eventuell bereits vorhandene Resultate.

Im folgenden wird jeder Parameter des Datenfeldes beschrieben. Die genannten Offsets sind Dezimalwerte, die sich auf den Beginn von ESTATE87 beziehen.

Offset 0: OPERATION ist ein Wortparameter, der den Fehlercode der Operation enthält, die das Exception verursachte. In Anlage 6 sind die Fehlercodes der WM87-Befehle enthalten. Anlage 16 beinhaltet die Fehlercodes der CEL87-Operationen. Der Fehlercode der DCON87-Programme ist immer 0C8H.

Offset 2: ARGUMENT ist ein Byteparameter, der Typ und Lage der Argumente der unterbrochenen Operation kennzeichnet. Das ARGUMENT-Byte ist wie folgt gegliedert:

	ATYPE2	PUSH ONCE	ATYPE1

Die 3-Bit-Felder ATYPE1 und ATYPE2 kennzeichnen den Typ der Argumente 1 und 2 gemäß der folgenden Codierung:

0 kein Argument
1 Argument auf ST0
2 weiteres Element auf ST1
3 Das Element des WM87-Stack wird durch REGISTER festgelegt.
4 eine Zahl im WM86-Speicher gemäß FORMAT
5 Das Argument ist eine Temporary-Real-Zahl.
6 Das Argument ist eine Long-Integer-Zahl.
7 Das Argument ist eine BCD-Zahl.

Das Bit PUSH ONCE ist gesetzt, wenn das Exception auf den WM87-Stack gebracht wurde. Das Bit ist

rückgesetzt, wenn ein oder zwei Argumente überschrieben wurden.

Offset 3: ARG1(5) ist ein Parameter, bestehend aus fünf Worten, die das erste Argument entsprechend dem unter ARGUMENT definierten Format enthalten.

Offset 13: ARG1_FULL ist ein Byteparameter, dessen niederwertiges Bit gesetzt ist, wenn Argument 1 existiert.

Offset 14: ARG2(5) ist ein Parameter, der aus fünf Worten besteht. Er enthält das zweite Argument entsprechend dem unter ARGUMENT definierten Wert.

Offset 24: ARG2_FULL ist ein Byteparameter, dessen niederwertiges Bit gesetzt ist, wenn Argument 2 existiert. Das Bit ist rückgesetzt, wenn nur ein Argument vorhanden ist, oder wenn der Exception-Handler gerufen wurde, nachdem die betreffende Operation beendet wurde.

Offset 25: RESULT ist ein Byteparameter, der Typ und Lage der Ergebnisse der unterbrochenen Operation kennzeichnet. Die Codierung der 3-Bit-Felder RTYPE1 und RTYPE2 entspricht derjenigen von ATYPE1 und ATYPE2. Die Bits POP ONCE und POP TWICE sind gesetzt, wenn die Operation den WM87-Stack ein- bzw. zweimal gepopt hat.

POP TWICE	RTYPE2	POP ONCE	RTYPE1

Offset 26: RES1(5) ist ein Parameter, bestehend aus fünf Worten, die das erste Resultat entsprechend dem in RESULT definierten Format enthalten.

Offset 36: RES1_FULL ist ein Byteparameter, dessen niederwertiges Byte gesetzt ist, wenn RES1 vorliegt. Wenn der Exception-Handler vor Ende der Operation aufgerufen wurde, ist noch kein Ergebnis vorhanden.

Offset 37: RES2(5) ist ein Parameter, bestehend aus fünf Worten. Sie enthalten das zweite Resultat in der unter RESULT festgelegten Form.

Offset 47: RES2_FULL ist ein Byteparameter, dessen niederwertiges Bit gesetzt ist, wenn RES2 vorliegt. Wenn nur RES1 vorliegt oder wenn der Exception-Handler vor Beendigung der Operation aufgerufen wurde, existiert RES2 nicht.

Offset 48: FORMAT ist ein Byteparameter, der den Datentyp spezifiziert, wenn ein 3-Bit-Feld in ARGUMENT oder RESULT den Wert 4 hat (Argument oder Resultat im Speicher). Die möglichen Werte von FORMAT sind:

0 bei short-real (32 Bit)
1 bei long-integer (64 Bit)
2 bei long-real (64 Bit)
3 bei word-integer (16 Bit)

Offset 49: REGISTER ist ein Byteparameter, der das Element des Register-Stack festlegt, wenn ein 3-Bit-Feld in ARGUMENT oder RESULT den Wert 3 hat. REGISTER kann Werte zwischen 0 (Stack-Top) und 7 annehmen.

Offset 50: SAVE87 ist ein aus 47 Worten bestehender Bereich. Er enthält den Status des WM87-Prozessors in der beim Befehl FSAVE festgelegten Form. Da DECODE aufgerufen wird, nachdem alle WM87-Exceptions gelöscht wurden, unterscheidet sich das Statuswort in SAVE87 von dem bei Auftreten des Trap. Der ursprüngliche Wert wird zu Beginn des Exception-Handlers in ERRORS87 abgelegt.

6.3.2. Regeln für die Nutzung der EH87

Durch Nutzung der Programme der EH87 kann eine Ortung und Behebung von WM87-Exception-Ursachen erfolgen, ohne daß der Nutzer spezielle Kenntnisse vom WM87-Prozessor besitzt. Das setzt aber eine genaue Befolgung der unten genannten Regeln voraus. Folgende Maßnahmen sind zu treffen:

- Der Exception-Handler muß eine Interrupt-Routine sein, die an den Interrupt 16 (00040H) anzuschließen ist.

- Wenn eine Rückkehr zum Programm, das den Fehler verursachte, erfolgen soll, müssen alle WM86-Register gerettet werden. Die WM86-Flags werden bei Interrupt-Routinen automatisch gerettet und wieder gesetzt.

- Der erste Gleitkommabefehl des Exception-Handlers muß FNSTSW sein, der das WM87-Statuswort in ein 16-Bit-Speicherwort bringt. Dieses Statuswort ist der Parameter ERRORS87 gemäß der EH87.

- Um die Synchronität mit dem Schaltkreis WM87 zu einem bestimmten Zeitpunkt zu sichern, muß es einen eigenständigen Zugriff zum WM86-Speicher geben. Dafür wird ein PUSH AX, gefolgt von einem POP AX empfohlen.

- Daran muß sich ein FNCLEX anschließen, um die WM87-Exceptions zu löschen. Die WM87-Exceptions müssen gelöscht sein, bevor EH87-Programme aufgerufen werden.

- Vor dem Aufruf von DECODE müssen alle entsprechenden Parameter auf den STACK gebracht werden.

- Falls NORMAL und SIEVE genutzt werden, müssen deren Aufrufe unmittelbar nach dem Aufruf von DECODE erfolgen, damit keine Änderung von ESTATE87 erfolgt. NORMAL und SIEVE erfordern, daß ESTATE87 durch DECODE definiert wurde. Wenn ESTATE87 keine Werte hat, die durch DECODE gesetzt wurden, sind die Ergebnisse von NORMAL und SIEVE undefiniert.

- An dieser Stelle muß der eigentliche Exception-Handler stehen.

- Für die Rückkehr aus dem Exception-Handler ist ENCODE zu nutzen.

- Die WM86-Register, die zu Beginn des Exception-Handlers gerettet wurden, sind zu aktualisieren.

- Der Exception-Handler muß mit einem IRET-Befehl verlassen werden.

Es ist zu beachten, daß alle Daten, die vom Exception-Handler benutzt werden, auf den WM86-Stack zu bringen sind, falls ein rekursiver Ruf zum Exception-Handler erfolgen kann. Das kann z. B. bei Wiederholung eines Befehls mit unmaskiertem Exceptionbit geschehen.

6.3.3. Linken der EH87

Wenn die Bibliotheken EH87 und CEL87 gelinkt werden sollen, ist darauf zu achten, daß im LINK-Kommando CEL87 vor EH87 erscheint. Das ist deshalb erforderlich, weil auch in EH87 alle Eintrittspunkte der CEL87 enthalten sind. Dadurch werden die ungenutzten CEL87-Operationen auf die jeweilige Pseudooperation in EH87 geführt, die nur einen HLT-Befehl enthalten. Auf diese Weise überschreitet die Länge des Programms nicht das unbedingt notwendige Maß.

Wenn die Reihenfolge im LINK-Kommando vertauscht wird, dann treten beim Linken keine Probleme auf. Die Abarbeitung von CEL87-Operationen ist aber nicht möglich, da die Programmverbindungen nicht zu CEL87 sondern zu EH87 erfolgten.
EH87 benötigt den Numerikprozessor. Demzufolge sind entweder die Interface-Bibliothek WM87 oder der Emulator EWM87 in Verbindung mit der Interface-Bibliothek EWM87.L86 im LINK-Kommando aufzuführen. Folgende Reihenfolge ist einzuhalten:

Anwenderprogramm
DCON87.L86 (wenn genutzt)
CEL87.L86 (wenn genutzt)
EH87.L86
WM87.L86 bzw. EWM87, EWM87.L86

6.4. Beschreibung von Teilprogrammen

6.4.1. DECODE

Das Programm DECODE liefert die Operation, die zum Trap führte, und rettet den Status des Prozessors.

Eingangsparameter

Vor dem Aufruf von DECODE sind ESTATE87_PTR und ERRORS87 auf den Stack zu bringen.
ESTATE87_PTR ist ein 4-Byte-Pointer, bestehend aus Segment und Offset, der auf ESTATE87 weist (siehe Abschnitt 6.3.1.). ERRORS87 ist als niederwertiges Byte eines Wortes auf den Stack zu bringen. Es entspricht dem niederwertigen Teil des Statuswortes zum Zeitpunkt nach Aufruf des Exception-Handlers und vor dem Rücksetzen der Exceptionbits. Das höherwertige Byte wird ignoriert.

Ausgangsparameter

Alle Ausgangsparameter werden auf den 144-Byte-Puffer ESTATE87 gebracht. Der interne Aufbau dieses Bereiches ist in Abschnitt 6.3.1. erläutert.
Der Prozessor wird auf den Status gesetzt, der von den

nachfolgenden Programmen der EH87 gefordert wird.

Funktion

DECODE liefert alle Informationen, die ein Exception-Handler zur Fehlerauswertung benötigt, auf dem Bereich ESTATE87. Dazu gehören der Status des Prozessors, die fehlerverursachende Operation sowie Formate und Werte von Argumenten und Resultaten. Die fehlerverursachende Operation kann sowohl ein WM87-Befehl als auch eine Routine von CEL87 bzw. DCON87 sein. Die Kennzeichnung der jeweiligen Operation ist in Anlage 6 bzw. 16 enthalten. Exceptions, die im Programm der DCON87 auftreten, werden in jedem Fall durch 0C8H gekennzeichnet.

Die Resultate der CEL87-Programme, die ihre Ergebnisse in WM86-Registern ablegen, werden durch DECODE auf den Register-Stack gebracht.
Nach Traps, die durch die Befehle FCOMP bzw. FCOMPP mit nicht normalisierten Argumenten verursacht wurden, stehen die Argumente bereits auf dem Register-Stack. DECODE bringt diese auf die aktuelle Register-Stack-Position.

6.4.2. ENCODE

ENCODE bereitet die Rückkehr in das trap-verursachende Programm vor, indem es den WM87-Prozessor in einen Zustand versetzt, der einer fehlerfreien Abarbeitung der betreffenden Operation entspricht.

Eingangsparameter

Vor dem Aufruf von ENCODE sind vier Parameter auf den Stack zu bringen. Die ersten beiden sind ESTATE87_PTR und ERRORS87 (siehe DECODE).
Der dritte Parameter ist CONTROL87. Er enthält den Wert, den das Steuerwort des Prozessors vor Wiederholung der ereignisverursachenden Operation (RETPY_FLAG = 1) annehmen soll.
Der vierte Parameter ist der Wortparameter RETRY_FLAG, durch dessen niederwertiges Bit festgelegt wird, ob die ereignisverursachende Operation wiederholt werden soll (RETRY_FLAG = 1) oder nicht.

Ausgangsparameter

Nach ENCODE befindet sich der WM87-Prozessor in einem ESTATE87 entsprechenden Zustand. Die Eingangsparameter sind vom Stack entfernt.

Funktion

ENCODE versetzt den WM87-Prozessor in einen Zustand, der dem nach einer korrekten Abarbeitung der trap-verursachenden Operation entspricht.
In Abhängigkeit vom RETRY_FLAG wurden zuvor unterschiedliche Maßnahmen eingeleitet.
Wenn RETRY_FLAG = 0 ist, dann wird angenommen, daß die Operation bereits durchgeführt wurde und Resultate vorliegen.
Wenn RETRY_FLAG = 1 ist, wird eine Wiederholung der trap-verursachenden Operation vorbereitet. Dazu werden die Argumente der Operation aktualisiert und das Steuerwort entsprechend

CONTROL87 gesetzt. Damit kann festgelegt werden, welche Exceptions zu Traps führen können. Wenn das Steuerwort unmaskierte Exceptions enthält, ist es möglich, daß während der Wiederholung einer Operation der Exception-Handler erneut aufgerufen wird. Um das zu verhindern, ist entweder das Exception zu maskieren (entsprechendes Bit in CONTROL87 setzen) oder die Argumente vor der Wiederholung der Operation zu verändern.
Wenn der zu wiederholende Befehl ein Transportbefehl ist, führt ENCODE die Operation mit maskierten Exceptions unter Nutzung der Argument- und Resultatwerte in ESTATE87 durch. Das ermöglicht es, ENCODE mit gesetztem RETRY_FLAG auch in den Fällen aufzurufen, in denen NORMAL "wahr" liefert, ohne befürchten zu müssen, daß sich das D-Exception wiederholt.
ENCODE kann nicht mit gesetztem RETRY_FLAG aufgerufen werden, wenn der Trap durch DCON87 verursacht wurde, da ESTAE87 nicht genügend Informationen über DCON87 enthält. Aus dem gleichen Grund kann ENCODE nicht die CEL87-Operationen mqerIC2, mqerIC4, mqerIE2 und mqerIE4 nach einem Genauigkeitsfehler wiederholen. Bei I-Exceptions kann ENCODE die Wiederholung der vier oben genannten Operationen sowie von mqerIA2 und mqerIA4 realisieren. Es ist jedoch zu beachten, daß die Ergebnisse nicht in den WM86-Registern, sondern im Register-Stack stehen.

6.4.3. FILTER

FILTER behandelt Unnormales und non-trapping-NaNs.

Eingangsparameter

Der Wortparameter ERRORS87 ist vor Aufruf von FILTER auf den Stack zu bringen (siehe DECODE).

Ausgangsparameter

Im niederwertigen Bit des AL-Registers steht das binäre Ergebnis von FILTER.
Liefert FILTER den Wert "falsch" (Carry-Flag = 0), dann wird der WM87-Prozessor auf den Stand vor Aufruf von FILTER gesetzt. Eine Fehlerbehandlung erfolgte nicht.
Liefert FILTER den Wert "wahr" (Carry-Flag = 1), so kann der Exception-Handler verlassen werden, da eine Fehlerbehandlung stattfand.

Funktion

FILTER bietet eine einfache Möglichkeit, die Leistungen der EH87 zu nutzen. Es ruft die Programme DECODE, SIEVE, NORMAL und ENCODE. Das Ergebnis von FILTER ist ein binärer Wert. Er liefert ein Aussage darüber, ob nach dem Aufruf von FILTER im Hauptprogramm fortgefahren werden kann, oder ob eine weitere Fehlerbehandlung erfolgen muß.
Liefert FILTER den Wert "wahr", dann wurde der Trap durch ein I- oder D-Exception verursacht. Der Fehler wurde korrigiert, und es kann im Hauptprogramm fortgesetzt werden.
Liefert FILTER den Wert "falsch", dann hat keine Fehlerbehandlung stattgefunden. Es müssen sich weitere Maßnahmen zur Fehlerermittlung und -korrektur anschließen.

6.4.4. NORMAL

NORMAL ermittelt und normalisiert nicht normalisierte Argumente.

Eingangsparameter

NORMAL erwartet, daß der WM87-Prozessor auf seinen Anfangsstatus gesetzt wurde. Informationen über den Status des Prozessors bei Auftreten des Traps sind in ESTATE87 enthalten.
Die Parameter ESTAE87_PTR und ERRORS87 sind vor dem Aufruf von NORMAL auf den Stack zu bringen (siehe Abschnitt 6.4.1.).

Ausgangsparameter

NORMAL liefert normalisierte Argumente im ESTATE87-Feld. Weiterhin liefert es einen binären Rückkehrcode im AL-Register.
Ist das niederwertige Bit im AL-Register gesetzt, so hat eine Normalisierung stattgefunden. Die trap-verursachende Operation kann durch den Aufruf von ENCODE mit gesetztem RETRY-Flag wiederholt werden.
Ist das niederwertige Bit im AL-Register rückgesetzt (Rückkehrcode falsch), dann wird keine Wiederholung gefordert.
NORMAL setzt den WM87-Prozessor auf den bei Aufruf vorgelegenen Status.

Funktion

NORMAL prüft zunächst, ob das D-Exception der einzige unmaskierte Fehler war. Liegen weitere Fehler vor, geht NORMAL mit dem Wert "falsch" in das aufrufende Programm zurück, um anzuzeigen, daß keine Wiederholung der unterbrochenen Operation gefordert wird.
Wenn ausschließlich ein unmaskiertes D-Exception vorliegt, liefert NORMAL den Rückkehrcode "wahr".
Wurde das D-Exception durch eine Ladeoperation verursacht, bleiben die unnormalisierten Argumente unverändert. Wenn die Operation kein Ladebefehl war, verändert NORMAL die unnormalisierten Argumente in ESTATE87. Short-Real- und Long-Real-Argumente werden normalisiert, Temporary-Real-Argumente auf Null gesetzt.
Wenn NORMAL den Rückkehrcode "wahr" liefert, muß ENCODE mit gesetztem RETRY_FLAG aufgerufen werden, um die trap-verursachende Operation zu wiederholen. Das Steuerwort kann dem bei Auftreten des Trap entsprechen (unmaskierte D-Exceptions). Bei Ladeoperationen erfolgt die Wiederholung durch ENCODE mit maskiertem D-Exception.
NORMAL liefert immer das Ergebnis "falsch", wenn der Trap durch DCON87 oder CEL87 verursacht wurde. Ein D-Exception kann nur durch einzelne WM87-Befehle hervorgerufen werden.
NORMAL rettet den Status des WM87-Prozessors.

6.4.5. SIEVE

SIEVE dient zur Ermittlung derjenigen NaNs, die keine weiteren Maßnahmen erfordern ("non-trapping"-NaNs).

Eingangsparameter

SIEVE setzt voraus, daß der WM87-Prozessor rückgesetzt ist. Alle Informationen über den Status bei Auftreten des Trap sind in ESTATE87 enthalten.
Vor dem Aufruf von SIEVE sind ESTATE87_PTR und ERRORS87 auf den Stack zu bringen (siehe DECODE).

Ausgangsparameter

Das Ergebnis von SIEVE ist ein binärer Wert, der im niederwertigen Bit des AL-Registers enthalten ist. Ist das Bit gesetzt (wahr), so kann das auftretende I-Exception ignoriert werden, da es durch ein "non-trapping"-NaN verursacht wurde (siehe Abschnitt 6.2.1.).
Der Status des WM87-Prozessors und ESTATE87 werden durch SIEVE nicht verändert.

Funktion

Beim Auftreten von I-Exceptions, die durch "non-trapping"-NaNs oder "non-ordered" comparison verursacht wurden, liefert SIEVE den Wert "wahr".
In diesen Fällen muß die offene Operation durch Aufruf von ENCODE mit gesetztem RETRY_FLAG und maskiertem I-Exception wiederholt werden. ENCODE stellt abschließend den Originalzustand des Steuerwortes (I-Exception unmaskiert) wieder her.
Liefert SIEVE den Wert falsch (AL = 0), so sind geeignete Maßnahmen zur Fehlerermittlung vorzunehmen.
Wenn das Exception durch eine DCON87- oder CEL87-Operation verursacht wurde, liefert SIEVE immer den Wert "falsch". Alle legalen "non-trapping" NaN-Exceptions rühren von einzelnen WM87-Befehlen her.
SIEVE folgt den in den Kapiteln "non-trapping"-NaNs und "non-ordered" comparison getroffenen Festlegungen.

Anlage 1 Datenarten des WM87

Datenart	Bit	Dezimal-stellen	Zahlenbereich
word-integer	16	4	-37768 ≤ x ≤ +32767
short-integer	32	9	-2*10**9≤ x ≤ +2*10**9
long-integer	64	18	-9*10**18 ≤x≤ +9*10**18
BCD	80	18	-99...99 ≤ x ≤ +99...99 (18 Stellen)
short-real	32	6-7	8,4 * 10 ** (-37) ≤ |x| |x| ≤ 3,37 * 10 ** 38
long-real	64	15-16	4,19 * 10 **(-307) ≤ |x| |x| ≤ 1,67 * 10 ** 308
temporary-real	80	19	3,4 * 10 **(-4932) ≤ |x| |x| ≤ 1,2 * 10 ** 4932

Anlage 2 Codierung in der Datenart integer

	Wert	Vorzeichen	signifikante Stellen
positiv	Maximum	0	11 ... 11
		.	. .
		.	. .
		.	. .
	Minimum	0	00 ... 01
	Null	0	00 ... 00
negativ	Minimum	1	11 ... 11
		.	. .
		.	. .
		.	. .
	Maximum *	1	00 ... 00
		word:	<---- 15 Bit ---->
		short	<---- 31 Bit ---->
		long	<---- 63 Bit ---->

* Der negative Maximalwert entspricht gleichzeitig dem Integer-Indefinite.

Anlage 3 Codierung in der Datenart BCD

Wert		Vorzei.		signifikante Stellen				
				Ziffer	Ziffer	Ziffer	...	Ziffer
positiv	Maximum	0	0000000	1001	1001	1001	...	1001
		.	.	.				.
	Minimum	0	0000000	0000	0000	0000	...	0001
	Null	0	0000000	0000	0000	0000	...	0000
negativ	Null	1	0000000	0000	0000	0000	...	0000
	Minimum	1	0000000	0000	0000	0000	...	0001
		.	.	.				.
	Maximum	1	0000000	1001	1001	1001	...	1001
Indefinite(*)		1	1111111	1111	1111	uuuu	...	uuuu

Zeichenerklärung

(*): Indefinite kann durch FBSTP gespeichert werden. Das Laden mittels FBLD erzeugt ein undefiniertes Resultat.

u: Die entsprechenden Bits sind undefiniert.

Anlage 4 Codierung in den Datenarten short- und long-real

Wert			Vorzeichen	biased exponent	signifikante Stellen	(*)
positive Werte		NaNs	0	11 ... 11	11 ...	11
			.	. .	. .	.
			0	11 ... 11	00 ...	01
		Infinity	0	11 ... 11	00 ...	00
	real	Normals	0	11 ... 10	11 ...	11
			.	. .	.	.
			0	00 ... 01	00 ...	00
		Denormals	0	00 ... 00	11 ...	11
			.	. .	.	.
			0	00 ... 00	00 ...	01
		Null	0	00 ... 0	00 ...	00
negative Werte	real	Null	1	00 ... 00	00 ...	00
		Denormals	1	00 ... 00	00 ...	01
			.	. .	.	.
			1	00 ... 00	11 ...	11
		Normals	1	00 ... 01	00 ...	00
			.	. .	.	.
			1	11 ... 10	11 ...	11
		Infinity	1	11 ... 11	00 ...	00
		NaNs	1	11 ... 11	00 ...	01
			:	: :	:	:
		Indefinite	1	11 ... 11	10 ...	00
		NaNs	1	11 ... 11	10 ...	01
			:	: :	:	:
			1	11 ... 11	11 ...	11
			short	<--8 Bit-->	<----- 23 Bit	----->
			long	<-11 Bit-->	<----- 52 Bit	----->

Anlage 4: (Fortsetzung)

Zeichenerklärung:

* Der gedachte Binärpunkt steht vor der ersten Bitposition.

Anlage 5 Codierung in der Datenart temporary-real

Wert			Vor- zeichen	biased Exponent	signifikante Stellen
positiv	NaNs		0 . . . 0	11 ... 11 . . . . . . 11 ... 11	111 ... 11 . . . . . . 100 ... 01
	Infinity		0	11 ... 11	100 ... 00
	real		0 0	11 ... 10 . . . . 00 ... 01	Normals 111 ... 11 . . . . 100 ... 00 Unnormals 011 ... 11 . . . . 000 ... 00
			0 . . 0	00 ... 00 . . . . 00 ... 00	Denormals 011 ... 11 . . . . 000 ... 01
		Null	0	00 ... 00	000 ... 00

Anlage 5: (Fortsetzung)

	Wert		Vorzeichen	biased Exponent	signifikante Stellen
n e g a t i v		Null	1	00 ... 00	000 ... 00
	real		1 . . . 1	00 ... 00 . . . . . . 00 ... 00	Denormals 000 ... 01 . . . . . . 011 ... 11
			1 . . . 1	00 ... 01 . . . . . . 11 ... 10	Unnormals 000 ... 00 . . . . 011 ... 11 Normals 100 ... 00 . . . . 111 ... 11
	Infinity		1	11 ... 11	100 ... 00
	NaNs		1 . . .	11 ... 11	100 ... 00 . . . . . .
		Indefinite	1	11 ... 11	110 ... 00
			. . . 1	. . . . . . 11 ... 11	. . . . . . 111 ... 11

Anlage 6 Befehle des Prozessors WM87

In Anlage 6 sind alle Befehle des WM87-Prozessors, deren Maschinencodes, Funktionen, Fehlercodes und möglichen Fehler aufgeführt. Im folgenden wird eine kurze Erläuterung der Darstellungsweise gegeben.

Befehlscode

In dieser Spalte erscheint der Maschinencode in der vom Assembler erzeugten Form. Ziffern und die Buchstaben A bis F stellen hexadezimale Zahlen dar. Die Zeichen i, j und / besitzen folgende Bedeutung:

- i beschreibt einen 4-Bit-Wert. Die letzten 3 Bit davon kennzeichnen das Register-Stack-Element des Befehls.
- j entspricht i + 8. Das erste Bit ist gesetzt. Die restlichen 3 Bit kennzeichnen das Register-Stack-Element des Befehls.
- / und eine nachfolgende Ziffer kennzeichnen das MODRM-Byte. (siehe Anleitung für den Programmierer) Die Ziffer entspricht dem REG-Feld im MODRM-Byte (Bits 3, 4, 5). Die Bits 6 und 7 (MOD-Feld) können 00, 01 und 10 sein. Das R/M-Feld (Bits 0, 1, 2) kann beliebige Werte annehmen. Bei einigen Werten von MOD und R/M gibt es ein oder zwei Verschiebebytes, die dem MODRM-Byte folgen.

Der Maschinencode für den WM87-Emulator wird beim Linken mit der Anschlußbibliothek EWM87.L86 erzeugt. Dadurch ergeben sich folgende Änderungen:

Der Befehl FWAIT wird in ein NOP(90H) überführt.
Alle anderen Befehle beginnen mit dem Interrupt-Code CDH. Wenn kein Segmentvorrangbyte existiert, wird aus dem zweiten Byte die Nummer des Interrupts gebildet.
Es ergeben sich Interrupt-Nummern von 18H bis 1FH. Wenn ein Segmentvorrangbyte vorhanden ist, wird es durch die Interrupt-Nummer ersetzt. Dabei gilt die folgende Zuordnung:

Segmentvorrang	Interrupt-Nummer
ES	14
CS	15
SS	16
DS	17

Alle weiteren Bytes werden nicht verändert. Sie werden durch die Interrupt-Routinen ausgewertet.

Befehl

Hier wird die Befehlsmnemonik mit den entsprechenden Operanden aufgeführt. Die Datenarten sind wie folgt gekennzeichnet:

r	real
i	integer
d	BCD

Funktion

Spalte 3 beinhaltet eine symbolische Darstellung des Befehls.

Anlage 6: (Fortsetzung)

Fehlercode(FC)

Spalte 4 enthält den hexadezimalen Wert des Fehlercodes. Er steht nach Ausführung des jeweiligen Befehls im Operationscoderegister.

Fehler

Hier sind alle bei dem jeweiligen Befehl möglichen Fehler aufgeführt.

Befehls-code	Befehl	Funktion	FC	Fehler
9B D9 F0	F2XM1	ST <-- (2**ST)-1	19	UP
9B D9 E1	FABS	ST <-- |ST|	01	I
9B DE C1	FADD	ST1 <-- ST1 + ST, pop	05	IDOUP
9B DC Ci	FADD STi,ST	STi <-- STi + ST	05	IDOUP
9B D8 Ci	FADD ST,STi	ST <-- STi + ST	05	IDOUP
9B D8 /0	FADD32 M	ST <-- ST + M32r	05	IDOUP
9B DC /0	FADD 64 M	ST <-- + M64r	05	IDOUP
9B DE Ci	FADDP STi,ST	STi <-- STi + ST, pop	05	IDOUP
9B DF /4	FBLD M	push,ST <-- M80d	10	I
9B DF /6	FBSTP M	M80d <-- ST,pop	0F	I
9B D9 E0	FCHS	ST <-- -ST	02	I
9B DB E2	FCLEX	clear exceptions		
9B D8 D1	FCOM	compare ST - ST1	03	ID
9B D8 Di	FCOM STi	compare ST - STi	03	ID
9B D8 /2	FCOM32 M	compare ST - M32r	03	ID
9B DC /2	FCOM64 M	compare ST - M64r	03	ID
9B D8 D9	FCOMP	compare ST - ST1,pop	03	ID
9B D8 Dj	FCOMP STi	compare ST - STi,pop	03	ID
9B D8 /3	FCOM32P M	compare ST - M32r,pop	03	ID
9B DC /3	FCOM64P M	compare ST - M64r,pop	03	ID
9B DE D9	FCOMPP	compare ST - ST1,pop2	03	ID
9B D9 F6	FDECSTP	decrememt stack pointer		
9B DB E1	FDISI	set interrupt mask		
9B DE F1	FDIV	ST1 <-- ST1/ST,pop	09	IDZOUP
9B DC Fj	FDIV STi,ST	STi <-- STi/ST	09	IDZOUP
9B D8 Fi	FDIV ST,STi	ST <-- ST/STi	09	IDZOUP
9B D8 /6	FDIV32 M	ST <-- ST/M32r	09	IDZOUP
9B DC /6	FDIV64 M	ST <-- ST/M64r	09	IDZOUP
9B DE Fj	FDIVP STi,ST	STi <-- STi/ST,pop	09	IDZOUP
9B DE F1	FDIVR	ST1 <-- ST/ST1,pop	09	IDZOUP
9B DC Fj	FDIVR STi,ST	Sti <-- ST/STi	0A	IDZOUP
9B D8 Fj	FDIVR ST,STi	ST <-- STi/ST	0A	IDZOUP
9B D8 /7	FDIVR32 M	ST <-- M32r/ST	0A	IDZOUP
9B DC /7	FDIVR64 M	ST <-- M64r/ST	0A	IDZOUP
9B DE Fi	FDIVRP STi,ST	STi <-- ST/STi,pop	0A	IDZOUP
9B DB E0	FENI	clear interrupt mask		
9B BD Ci	FFREE STi	empty STi		
9B DE /0	FIADD16 M	ST <-- ST + M16i	05	IDOP
9B DA /0	FIADD32 M	ST <-- ST + M32i	05	IDOP
9B DE /2	FICOM16 M	compare ST - M16i	03	ID
9B DA /2	FICOM32 M	compare ST - M32i	03	ID
9B DE /3	FICOM16P M	compare ST - M16i,pop	03	ID

Anlage 6: (Fortsetzung)

Befehls-code	Befehl	Funktion	FC	Fehler
9B DA /3	FICOM32P M	compare ST - M32,pop	03	ID
9B DE /6	FIDIV16 M	ST <-- ST/M16i	09	IDZOUP
9B DA /6	FIDIV32 M	ST <-- ST/M32i	09	IDZOUP
9B DE /7	FIDIVR16 M	ST <-- M16i/ST	0A	IDZOUP
9B DA /7	FIDIVR32 M	ST <-- M32i/ST	0A	IDZOUP
9B DF /0	FILD16 M	push,ST <-- M16i	10	I
9B DB /0	FILD32 M	push,ST <-- M32i	10	I
9B DF /5	FILD64 M	push,ST <-- M64i	10	I
9B DE /1	FIMUL16 M	ST <-- ST * M16i	08	IDOP
9B DA /1	FIMUL32 M	ST <-- ST * M32i	08	IDOP
9B D9 F7	FINCSTP	increment stack pointer		
9B DB F3	FINIT	initialisiere WM87		
9B DF /2	FIST16 M	M16i <-- ST	0F	IP
9B DB /2	FIST32 M	M32i <-- ST	0F	IP
9B DF /3	FIST16P M	M16i <-- ST,pop	0F	IP
9B DB /3	FIST32P M	M32i <-- ST,pop	0F	IP
9B DF /7	FIST64P M	M64i <-- ST,pop	0F	IP
9B DE /4	FISUB16 M	ST <-- ST - M16i	06	IDOP
9B DA /4	FISUB32 M	ST <-- ST - M32i	06	IDOP
9B DE /5	FISUBR16 M	ST <-- M16i - ST	07	IDOP
9B DA /5	FISUBR32 M	ST <-- M32i - ST	07	IDOP
9B D9 Ci	FLD STi	push,ST <-- old STi	10	I
9B DB /5	FLD80 M	push,ST <-- M80r	10	ID
9B D9 /0	FLD32 M	push,ST <-- M32r	10	ID
9B DD /0	FLD64 M	push,ST <-- M64r	10	ID
9B D9 E8	FLD1	push,ST <-- 1.0	11	I
9B D9 /5	FLDCW M	Steuerwort <-- M16i		
9B D9 /4	FLDENV M	Umgebung <-- M112		
9B D9 EA	FLDL2E	push,ST <-- logbase 2 of e	13	I
9B D9 E8	FLDL2T	push,ST <--logbase 2 of 10	12	I
9B D9 EC	FIDLG2	push,ST <--logbase 10 of 2	15	I
9B D9 ED	FLDLN2	push,ST <--logbase e of 2	16	I
9B D9 EB	FLDPI	push,ST <-- PI	14	I
9B D9 EE	FLDZ	push,ST <-- +0.0	17	I
9B DE C9	FMUL	ST1 <-- ST1 * ST,pop	08	IDOUP
9B D8 Cj	FMUL ST,STi	ST <-- ST * STi	08	IDOUP
9B DC Cj	FMUL STi,ST	STi <-- STi * ST	08	IDOUP
9B D8 /1	FMUL32 M	ST <-- ST * M32r	08	IDOUP
9B DC /1	FMUL64 M	St <-- ST * M64r	08	IDOUP
9B DE Cj	FMULP STi,ST	ST <-- STi * ST,pop	08	IDOUP
90 DP E2	FNCLEX	nowait, clear exceptions		
90 DB E1	FNDISI	nowait, set interrupt mask		
90 DB E0	FNENI	nowait, clear interrupt mask		
90 DB E3	FNINIT	nowait, initialize WM87		
90 D9 D0	FNOP	no, operation		
90 DD /6	FNSAVE M	nowait, M752 <-- WM87 state		
90 D9 /7	FNSTCW M	nowait, M16<--control word		
90 D9 /6	FNSTENV M	nowait, M112<--environment		
90 DD /7	FNSTSW M	nowait, M16 <-- status word		

Anlage 6: (Fortsetzung)

Befehls-code	Befehl	Funktion	FC	Fehler
9B D9 F3	FPATAN	ST <-- arctan(ST1*ST),pop	1D	UP
9B D9 F8	FPREM	ST <-- REPEAT(ST - ST1)	1E	IDU
9B D9 F2	FPTAN	push,ST1/ST<-- tan(old ST)	1C	IP
9B D9 FC	FRNDINT	ST <-- round(ST)	1F	IP
9B DD /4	FRSTOR M	WM87state <-- M752		
9B DD /6	FSAVE M	M752 <-- WM87state		
9B D9 FD	FSCALE	ST <-- ST * 2 ** ST1	18	IOU
9B D9 FA	FSQRT	ST <-- square root	0C	IDP
9B DD Di	FST STi	STi <-- ST	0F	I
9B D9 /2	FST32 M	M32r <-- ST	0F	IOUP
9B DD /2	FST64 M	M64r <-- ST	0F	IOUP
9B D9 /7	FSTCW M	M16i <-- control word		
9B D9 /6	FSTENV M	M112 <-- environment		
9B DD Dj	FSTP STi	STi <-- ST,pop	0F	I
9B DB /7	FST80P M	M80r <-- ST,pop	0F	I
9B D9 /3	FST32P M	M32r <-- ST,pop	0F	IOUP
9B DD /3	FST64P M	M64r <-- ST,pop	0F	IOUP
9B DD /7	FSTSW M	M16i <-- status word	0F	IOUP
9B DE E9	FSUB	ST1 <-- ST1 - ST,pop	06	IDOUP
9B DC Ej	FSUB STi,ST	STi <-- STi - ST	06	IDOUP
9B D8 Ei	FSUB ST,STi	ST <-- ST - STi	06	IDUOP
9B D8 /4	FSUB32 M	ST <-- ST - M32r	06	IDOUP
9B DC /4	FSUB64 M	ST <-- ST - M64r	06	IDOUP
9B DE Ej	FSUBP STi,ST	STi <-- STi - ST,pop	06	IDOUP
9B DE E1	FSUBR	ST1 <-- ST - ST1,pop	07	IDOUP
9B DC Ei	FSUBR STi,ST	STi <-- ST - STi	07	IDOUP
9B D8 Ej	FSUBR ST,STi	ST <-- STi - ST	07	IDOUP
9B D8 /5	FSUBR32 M	ST <-- M32r - ST	07	IDOUP
9B DC /5	FSUBR64 M	ST <-- M64r - ST	07	IDOUP
9B DE Ei	FSUBRP STi,ST	STi <-- ST - STi,pop	07	IDOUP
9B D9 E4	FTST	compare ST - 0.0	04	ID
9B	FWAIT	wait for WM87 ready		
9B D9 E5	FXAM	C3 - C0 <-- type of ST		
9B D9 C9	FXCH	exchange ST and ST1	0B	I
9B D9 Cj	FXCH STi	exchange ST and STi	0B	I
9B D9 F4	FXTRACT	push,ST1 <-- expo,ST <--sign	0B	I
9B D9 F1	FYL2X	ST <-- ST1 * log 2(ST),pop	1A	P
9B D9 F9	FYL2XP1	ST <-- ST1 * log 2(ST+1),pop	1B	P

Anlage 7 Status des Prozessors nach Initialisierung durch INIT87

	Wert	Bedeutung
Steuerwort		
Infinity Control	0	Projektiv-Modus
Runding Control	00	Rundung zum nächsten Wert
Precision Control	11	64 Bit
Interrupt-Enable-Mask	1	Interrupt verboten
Exception-Mask	111111	alle Exception maskiert
Statuswort		
Busy	0	nicht aktiv
Condition-code	????	undefiniert
Stack-Top	000	Stack ist leer
Interrupt-Request	0	keine Interruptanforderung
Exception-Flags	000000	keine Exceptions
Zustandswort		
Tags	11	leer
Register		unverändert
Exception-Pointers		
Instruction-Code		unverändert
Instruction-Adresse		unverändert
Operanden-Adresse		unverändert

Anlage 8 Reaktion des Prozessors auf maskierte Exceptions

Ursache	Wirkung
invalid operation	
Quellregister ist als leer gekennzeichnet (Tagwort = 11B) (Stack-Underflow)	real-indefinite
Zielregister ist nicht als leer gekennzeichnet (Stack-Overflow)	real-indefinite (Überschreiben des Zielwertes)
ein oder beide Operanden sind NaN	NaN mit größerem absoluten Wert
ein oder beide Operanden sind NaN (nur bei Vergleichs- und Testoperationen)	Bedingungscode "not comparable" gesetzt
Im Affin-Modus sind beide Operanden Infinities mit entgegengesetztem Vorzeichen. Im Projektiv-Modus sind beide Operanden Infinities (nur bei Addition).	real-indefinite
Im Affin-Modus haben beide Operanden gleiches Vorzeichen oder im Projektiv-Modus sind beide Operanden Infinities. (nur bei Subtraktion)	real-indefinite
+Infinity * 0 oder -Infinity * 0 (nur bei Multiplikation)	real-indefinite
Infinity/Infinity, 0/0, 0/Pseudonull Divisor de- oder unnormal (nur bei Division)	real-indefinite
Divisor de- oder unnormal oder Dividend Indefinite (nur bei FPREM)	real-indefinite Bedingungscode "complete remainder" gesetzt
Operand ist ungleich 0 oder negativ oder Operand ist de- bzw. unnormal oder im Affin-Modus ist der Operand -Infinity oder im Projekt-Modus +Infinity (nur bei FSQRT)	real-indefinite

Anlage 8: (Fortsetzung)

Ursache	Wirkung
Im Projektiv-Modus wird +Infinity verglichen mit 0, +Infinity oder normalisierter Zahl (nur bei Vergleichsoperationen)	Bedingungscode "not comparable" gesetzt
Im Projektiv-Modus ist der Operand Infinity. (nur bei FTST-Befehl)	Bedingungscode "not comparable" gesetzt.
Quellregister ist leer, ein NaN, de- bzw. unnormal, +Infinity oder >18 Dezimalstellen. (nur bei FBSTP-Befehl)	BCD-Infinity wird abgespeichert.
Quellregister ist leer, ein NaN, de- bzw. unnormal, Infinity oder nicht darstellbar. (nur bei FIST, FISTP-Befehlen)	integer-indefinite wird abgespeichert.
Ziel ist short- oder long-real und im Quellregister ist ein Unnormal mit zulässigem Exponenten. (nur bei FST, FSTP-Befehlen)	real-indefinite wird abgespeichert.
Ein oder zwei Register sind als leer gekennzeichnet. (nur bei FXCH)	Das leere Register wird real-indefinite gesetzt und dann gewechselt.
denormalisierter Operand	
Quelloperand ist denormalisiert (nur bei FLD)	wird geladen
Ein oder zwei Operanden sind denormalisiert. (nur bei arithmetischen Operationen)	zu unnormal konvertiert und Operation durchgeführt
Ein oder zwei Operanden sind de- oder unnormalisiert, nicht Pseudonull. (nur bei Vergleichs- und Testoperationen)	eventuell zu Unnormal konvertiert, normalisiert und Operation durchgeführt.

Anlage 8: (Fortsetzung)

Ursache	Wirkung
Division durch Null	
Divisor gleich 0 (nur bei Division)	Liefert Infinity mit Vorzeichen, das sich aus "exclusivem Oder" der Operandenvorzeichen ergibt.
Overflow	
Rundungsmodus ist nearest oder chop und der Exponent größer als 16383. (nur bei arithmetischen Operationen)	Infinity mit Vorzeichen gemäß den Operanden, Ereignisbit "Precision" ist gesetzt.
Rundungsmodus ist nearest oder chop, der Exponent ist größer als +127 bei short-real bzw. +1023 bei long-real. (nur bei FST, FSTP-Befehlen)	Infinity mit Vorzeichen gemäß den Operanden, Exceptionbit "Precision" ist gesetzt.
Underflow	
Der Exponent des Resultates ist kleiner -16382. (nur bei arithmetischen Operationen)	Denormalisieren bis der Exponent gleich -16382. Runden der Mantisse auf 64 Bit, wenn Mantisse = 0, dann 0 als Ergebnis; ansonsten Denormal.
Ziel ist long-real und der Exponent ist kleiner -1022. (nur bei FST, FSTP-Befehlen)	Denormalisieren bis der Exponent -1022 ist. Runden der Mantisse auf 53 Bit. Wenn Mantisse = 0 dann wird 0 abgespeichert, ansonsten Denormal (binärer Exonent = 0).
Zielformat ist short-real und der Exponent ist kleiner -126. (nur bei FST-, FSTP-Befehlen)	Denormalisieren bis der Exponent -126 ist. Runden der Mantisse auf 24 Bit. Wenn Mantisse = 0, dann wird 0 abgespeichert, ansonsten Denormal (binärer Exponent = 0).

Anlage 8: (Fortsetzung)

Ursache	Wirkung
Precision	
Rundungsbit ist im Statuswort gesetzt.	keine Reaktion

Anlage 9 Maskierte Reaktion bei Rundung mit Überlauf

Ergebnisse vor dem Runden		Rundungs-Modus	Ergebnis
Mantisse	Vorz.		
normalisiert	+	aufrunden	+Infinity
normalisiert	+	abrunden	größte positive Zahl 1)
normalisiert	−	aufrunden	größte negative Zahl 1)
normalisiert	−	abrunden	−Infinity
unnormalisiert	+	aufrunden	+Infinity
unnormalisiert	−	abrunden	größter Exponent, Mantisse unverändert 2)
unnormalisiert	+	aufrunden	größter Exponent, Mantisse unverändert 2)
unnormalisiert	−	abrunden	−Infinity

1) Die größten erlaubten Real-Werte sind:

Exponent: 11 ... 10B
Mantisse: (1).11 ... 10B

2) Die Mantisse bleibt unnormalisiert. Die Rundung erfolgt wie gewöhnlich (mit "chopped toward" 0). Der Exponent lautet 11 ... 10B.

Anlage 10 Reaktion der Prozessoren auf die maskierten und unmaskierten Exceptions

Exception	maskiert	unmaskiert
Invalid Operation	NaN: eine oder beide Operationen sind NaN Indefinite: in allen anderen Fällen	Interrupt 16
Zerodivide	Indefinite mit dem Vorzeichen, das sich aus exclusivem Oder der Operandenvorzeichen ergibt.	Interrupt 16
Denormalized	Speicheroperanden werden wie üblich behandelt, Registeroperanden werden in Unnormales überführt.	Interrupt 16
Overflow	liefert vorzeichenbehaftetes Infinity	bei Speicheroperand: Interrupt 16 bei Registeroperand: Modifikation des Exponenten *) Interrupt 16
Underflow	denormalisierter Ergebniswert	bei Speicheroperand: Interrupt 16 bei Registeroperand: Modifikation des Exponenten *) Interrupt 16
Precission	gerundeter Ergebniswert	gerundetes Ergebnis Interrupt 16

*) Bei Overflow wird 24576 vom Exponenten des Ergebnisses subtrahiert. Dadurch liegt der Exponent wieder im gültigen Bereich. Im Exception-Handler kann diese Korrektur berücksichtigt werden.
Bei Underflow wird die Konstante 24576 zum Ergebnisexponenten addiert.

Anlage 11 Ergebnisse bei Infinity-Operanden

Operation	Ergebnis	
	Projektiv-Modus	Affin-Modus
Addition		
+Inf plus +Inf	invalid operation	+Inf *)
-Inf plus -Inf	invalid operation	-Inf
+Inf plus -Inf	invalid operation	invalid operation
-Inf plus +inf	invalid operation	invalid operation
±Inf plus ±x	Inf 1)	Inf 1)
±x plus ±Inf	Inf 1)	Inf 1)
Subtraktion		
+ Inf minus - Inf	invalid operation	+ Inf
- Inf minus + inf	invalid operation	- Inf
+ Inf minus + Inf	invalid operation	invalid operation
- Inf minus - Inf	invalid operation	invalid operation
±Inf minus ± x	Inf 1)	Inf 1)
± x minus ±Inf	Inf 2)	Inf 2)
Multiplikation		
±Inf mal ± Inf	Inf 3)	
±Inf mal ± y	Inf 3)	
± 0 mal ± Inf	invalid operation	invalid operation
± 0 mal ± 0	invalid operation	invalid operation
Division		
±Inf durch ±Inf	invalid operation	invalid operation
±Inf durch ± x	Inf 3)	Inf 3)
± x durch ±Inf	0 3)	0 3)
FSQRT		
-Inf	invalid operation	invalid operation
+Inf	invalid operation	+Inf
FPREM		
±Inf rem ±Inf	invalid operation	
±Inf rem ± x	invalid operation	
± y rem ±Inf	y 1)	y 1)
± 0 rem ±Inf	0 1)	0 1)
FRNDINT		
± Inf	Inf 1)	Inf 1)
FSCALE		
±Inf scaled by ±Inf	invalid operation	invalid operation
±Inf scaled by ± x	Inf 1)	Inf 1)
± 0 scaled by ±Inf	0 1)	0 1)
± y scaled by ±Inf	invalid operation	invalid operation

Anlage 11: (Fortsetzung)

Operation	Ergebnis	
	Projektiv-Modus	Affin-Modus
FXTRACT		
± Inf	invalid operation	invalid operation
Compare *		
±Inf : ±Inf	A = B	-Inf < +Inf
±Inf : ± y	invalid operation	-Inf < y < +Inf
±Inf : ± 0	invalid operation	-Inf < 0 < +Inf
FTST		
± Inf	invalid operation	Inf 1)

Zeichenerklärung

x - Operand ist Null oder ungleich Null
y - Operand ist ungleich Null
1) - Vorzeichen des ursprünglichen Operanden
2) - Vorzeichen ist verschieden von dem des ursprünglichen Operanden
3) - Vorzeichen ergibt sich aus exclusivem Oder der Operandenvorzeichen, d. h. + bei Übereinstimmenden, - bei unterschiedlichem Vorzeichen der Operanden.
*) - Inf ist eine Abkürzung und steht für Infinity.

Anlage 12 Ergebnisse bei Null-Operanden

Operation/Operanden	Ergebnis	Operation/Operanden	Ergebnis
FLD, FBLD (1)		FCMS	
+0	+0	+0	−0
−0	−0	−0	+0
FILD (2)		FTST	
+0	+0	±0	Null
FS, FSTP		FABS	
+0	+0	±0	+0
−0	−0		
+x (3)	+0		
−x (3)	−0	F2XM1	
		+0	+0
FIST, FISTP		−0	−0
+0	+0		
−0	+0	FRNDINT	
+x (4)	+0		
−x (4)	+0	+0	+0
		−0	−0
Compare			
		FXTRACT	
±0 : +x	A < B		
±0 : ±0	A = B	+0	beide +0
±0 : −x	A > B	−0	beide −0
FSQRT			
−0	−0		
+0	+0		

Bemerkungen:

(1) Arithmetische und Vergleichsoperationen mit realen Speicheroperanden interpretieren das Vorzeichen auf die gleiche Weise.
(2) Arithmetische und Vergleichsoperationen mit Integer-Operanden interpretieren das Vorzeichen auf die gleiche Weise.
(3) Underflows können beim Speichern Null ergeben.
(4) Kleine Werte (|x| < 1) können beim Speichern von Integer-Zahlen Null ergeben.

Anlage 12: (Fortsetzung)

Operation/Operanden	Ergebnis
Addition	
+0 plus +0	+0
−0 plus −0	−0
+0 plus −0, −0 plus +0	(5)
−x plus +x, +x plus −x	(5)
±0 plus ±x, ±x plus ±0	#x (6)
Subtraktion	
+0 minus −0	+0
−0 minus +0	−0
+0 minus +0, −0 minus −0	0 (5)
+x minus +x, −x minus −x	0 (5)
±0 minus ±x, ±x minus ±0	#x (6)
Multiplikation	
+0 mal +0, −0 mal −0	+0
+0 mal −0, −0 mal +0	−0
+0 mal +x, +x mal +0	+0
+0 mal −x, −x mal +0	−0
−0 mal +x, +x mal −0	−0
−0 mal −x, −x mal −0	+0
+x mal +y, −x mal −y	+0, Underflow (7)
+x mal −y, −x mal +y	−0, Underflow (7)
Division	
±0 durch ±0	invalid operation
±x durch ±0	Zerodivide
+0 durch +x, −0 durch −x	+0
+0 durch −x, −0 durch +x	−0
−x durch −y, +x durch +y	+0, Underflow (7)
−x durch +y, +x durch −y	−0, Underflow (7)
FPREM	
±0 rem ±0	invalid operation
±x rem ±0	invalid operation
+0 rem +x, +0 rem −x	+0
−0 rem +x, −0 rem −x	−0
+x rem +y, +x rem −y	+0 (8)
−x rem −y, −x rem +y	−0 (8)

Bemerkungen:

(5) Das Vorzeichen wird durch den Rundungsmodus festgelegt.
+: Runden auf nächsliegenden Wert; Aufrunden oder kein Runden
−: Abrunden

(6) # = Vorzeichen von x

(7) Sehr kleine Werte von x und y können nach dem Runden des Resultats Null ergeben. Der Prozessor zeigt in diesem Fall einen Underflow an.

(8) wenn x ohne Rest durch y teibar ist

Anlage 13 Ergebnisse bei unnormalisierten Operanden (Unormals)

Operation	Ergebnis
Addition/Subtraktion	Der größte absolute Wert bestimmt das Ergebnis.
Multiplikation	Bei unnormalisierten Operanden ist das Ergebnis unnormalisiert.
Division (nur unnormalisierter Dividend)	Ergebnis ist unnormalisiert.
FPREM (nur unnormalisierter Dividend)	Ergebnis ist normalisiert.
Division/FPREM (unnormalisierter Divisor)	invalid operation
Compare /FTST	wenn möglich, Normalisierung vor der Operation
FRNDINT	wenn möglich, Normalisierung vor der Operation
FSQRT	invalid operation
FST, FSTP (short/long-real destination)	Liegt der Wert oberhalb des Grenzwertes, der einen Underflow erzeugen würde, wird Underflow angezeigt, ansonsten invalid operation.
FSTP (temporary-real destination)	gespeichert wie gewöhnlich
FIST, FISTP, FBSTP	invalid operation
FLD	geladen wie gewöhnlich
FXCM	Austausch wie gewöhnlich
transzendente Operationen	undefiniert, Operanden werden nicht geprüft

Anlage 14 Datenfeld ESTATE87 der EH87

ESTATE87:

OPERATION	WORD	Befehl oder Funktion, die das Exception verursachte
ARGUMENT	BYTE	zwei Tetraden, codiert in folgender Form: 0 - kein Operand 1 - ST0 2 - ST1 3 - ST(REGISTER) 4 - siehe FORMAT 5 - temporary-real 6 - 64 Bit integer 7 - BCD Bit 3 heißt: 1x push
ARG1(5)	WORD	Parameter entsprechend der niederwertigen Tetrade in ARGUMENT
ARG1_FULL	BYTE	gesetzt, wenn ARG1 einen Wert enthält
ARG2(5)	WORD	Parameter entsprechend der höherwertigen Tetrade in ARGUMENT
ARG2_FULL	BYTE	gesetzt, wenn ARG2 einen Wert enthält
RESULT	BYTE	Format des Resultats entsprechend ARGUMENT, außer: Bit 3 bedeutet 1x pop, Bit 7 bedeutet 2x pop
RES1(5)	WORD	Wert des Resultats entsprechend der niederwertigen Tetrade in RESULT
RES1_FULL	BYTE	gesetzt, wenn RES1 einen Wert enthält
RES2(5)	WORD	Wert des Resultats entsprechende der höherwertigen Tetraden in RESULT
RES2_FULL	BYTE	gesetzt, wenn RES2 einen Wert enthält
FORMAT	BYTE	Format des Typs 4 von ARGUMENT oder RESULT 0 - 32 Bit real 1 - 32 Bit integer 2 - 64 Bit real 3 - 16 Bit integer
REGISTER	BYTE	Register-Stack-Nummer für Typ 3 von ARGUMENT oder RESULT
CONTROL_WORD	WORD	WM87-Steuerwort
STATUS_WORD	WORD	WM87-Statuswort
TAG_WORD	WORD	WM87-Tagwort
ERROR_POINTERS(5)	WORD	WM87-Befehlszähler, Befehlscode, Operand, Pointer
STACK_87(40)	WORD	WM87-Stack der 8 temporären Real-Werte

Anlage 15 Globale Symbole

Neben den Programmnamen sind innerhalb der Bibliotheken weitere globale Symbole definiert. Eine Doppelvereinbarung im Anwenderprogramm ist zu vermeiden.
Die Übersetzter erzeugen beim Auftreten von WM87-Befehlen im Anwenderprogramm Referenzen zu den globelen Symbolen der Interface-Bibliotheken. Eine Doppelvereinbarung dieser speziellen Globals ist nicht möglich.

Interface-Bibliotheken

M:_NCS
M:_NDS
M:_NES
M:_NSS
M:_NST
M:_WCS
M:_WDS
M:_WES
M:_WSS
M:_WST
M:_WT

DCON87.L86

CHK_UNMSKD_O_U_ERR
MQCBINDEC
MQBIN_DECLOW
MQCDBX
MQCDBXDB
MQCDECBIN
MQCDECBINLO
MQCDECLOW_BIN
MQCDEC_BIN
MQCLONG_TEMP
MQCSHORT_TEMP
MQCSNGXDB
MQCSNX
MQCTEMP_LONG
MQCTEMP_SHORT
MQCXDB
MQCXDBDB
MQCXDBSNG
MQCXSN
POWER_OF_IO
UNMSKD_OV_OR_UN
XCPTN_RTRN

Anlage 15: (Fortsetzung)

CBL87.L68

MQERACS	MQERMIN	MQ_DECIDE
MQERAIN	MQERMOD	MQ_EXIT
MQERANT	MQERN12	MQ_EXMI
MQERASN	MQERNIN	MQ_I
MQERAT2	MQERR12	MQ_IRCHK
MQERATN	MQERRMD	MQ_LOG
MQERC12	MQERRNT	MQ_LOG10
MQERCOS	MQERSGN	MQ_LOGDN
MQERCSH	MQERSIN	MQ_MQRPI
MQERDIM	MQERSNH	MQ_NAN
MQEREXP	MQERTAN	MQ_NOF
MQERIA2	MQERTNH	MQ_NORM
MQERIA4	MQERY2X	MQ_NQ
MQERIAX	MQERY12	MQ_OF
MQERIC2	MQERY14	MQ_PO
MQERIC4	MQERYIS	MQ_P12
MQERICX	MQ_1	MQ_P11
MQERIE2	MQ_2XM1	MQ_Q
MQERIE4	MQ_63U	MQ_RAD
MQERIEX	MQ_63U1	MQ_RERR
MQERINT	MQ_63UP12	MQ_SIN
MQERIRT	MQ_AT2	MQ_TXAM
MQERLGD	MQ_CONST	MQ_UO
MQERLGE	MQ_COS	MQ_YL2X
MQERMAX	MQ_CP2N63	

Anlage 15: (Fortsetzung)

EH87.L68

DECODE	MQERSGN
ENCODE	MQERSIN
FILTER	MQERSNH
FQORTRANSATAUSCHECK	MQERTAN
MQERACS	MQERTNH
MQERAIN	MQERY2X
MQERANT	MQERY14
MQERSAN	NORMAL
MQERAT2	SIEVE
MQERATN	TQDECODE87
MQERC12	TQENCODE87
MQERCOS	TQFETCH_AND_STORE
MQERCSH	TQINSTRUCTION_RETRY
MQERDIM	TQNANFILTER
MQEREXP	TQNORM87
MQERINT	TQNORMALIZE
MQERIRT	TQPOP_THE_TOP
MQERLGD	TQREALMATHFILTER
MQERLGE	TQRESTORE_PTRS
MQERMAX	TQSAVE_PTRS
MQERMIN	TQUNPOP_THE_TOP
MQERMOD	TQ_312
MQERN12	TQ_320
MQERNIN	TQ_322
MQERR12	TQ_324
MQERRMD	TQ_326
MQERRNT	

Anlage 16 Programme der CEL87.L86

Die folgende Zusammenstellung enthält die Namen der CEL87-Programme, eine symbolische Funktionsbeschreibung sowie Fehlercodes und Fehlerkennzeichen.
x und y kennzeichnen die Werte auf den Register-Stack-Pointern ST und ST1. STk ist ein im WM86-Stack enthaltener Wortparameter.
Beim Aufruf des Exception-Handlers stehen in Abhängigkeit von der Art des verursachenden Exceptions die Eingangs- oder die Ausgangsparameter im Register-Stack. Dementsprechend sind die Fehlerkennzeichen der jeweiligen Spalte zugeordnet.
Die erste Ziffer des Fehlercodes kennzeichnet die Anzahl der Parameter im Register-Stack.
Bei den Programmen mqerDIM, mqerMAX, mqerMIN und mqerSGN können D-Exceptions vor dem Programmende auftreten. Der Exception-Handler muß die de- bzw. unnormalisierten Werte korrigieren und in dem jeweiligen Programm fortsetzen.

Name	Funktion	Fehlercode, Stack enthält	
		Eingangswert	Ausgangswert
mqerACS	x = arccosin(x)	175 I	
mqerASN	x = arcsin(x)	174 I	
mqerAT2	x = arctangent(x)	277 I	277 U
mqerATN	x = arctangent(x)	176 I	
mqerCOS	x = cosine(x)	172 I	
mqerCSM	x = hypercosine(x)	16F IO	
mqerDIM	x = max(y-x,+0)	265 I	265 OU
mqerEXP	x = e (3)	16B IOU	
mqerIA2	AX = roundaway(x)	17E I	
mqerIA4	DXAX = rundaway(x)	168 I	
mqerIAX	x = roundaway(x)	167 I	
mqerIC2	AX = chop(x)	17E I	17E P
mqerIC4	DXAX = chop(x)	179	I
mqerICX	x = chop(x)	166 I	166 P
mqerIE2	AX = roundeven(x)	180 I	180 P
mqerIE4	DXAX = roundeven(x)	17B I	17B P
mqerIEX	x = roundeven(x)	178 I	178 P
mqerLGD	x = common log(x)	16D IZ	
mqerLGD	x = natural log(x)	16C IZ	
mqerMAX	x = max(x,y)	282 I	
mqerMIN	x = min(x,y)	281 I	
mqerMOD	x = (y mod x),has sign(y)	269 I	269 U
mqerRMD	x = (y mod x), close to 0	27A I	27A U
mqerSGN	x = (y with x's sign)	264 I	
mqerSIN	x = sine(x)	171 I	
mqerSNH	x = hyperpolic sine(x)	16E IO	
mqerTAN	x = tangent(x)	173 IZ	
mqerTNH	x = hyperpolic tangent(x)	170 I	
mqerY2X	x = y ** x	26A IZOU	
mqerYI2	x = x ** AX	27C IOU	
mqerYI4	x = x ** DXAX	27C IOU	
mqerYIS	x = x ** STk	27C IOU	

Sachwortverzeichnis

Befehlssatz 8, 12

D-Exception 29f
Darstellung, extern 19
Datenart 8, 19
Datenformat 8
Denormal 11
Dezimalpunkt 24
Dezimalzahl 25f
Dezimalziffer 24, 27

Emulator 16, 18
Exception 13, 15, 21, 36
Exception-Handler 16, 21, 26, 78
Exponent 23

Fehlercode 79

Genauigkeit 20
Genauigkeitsverlust 20
Gleitkommaformat 19
Gleitkommazahl 19, 24, 28

I-Exception 31
Indefinity 12
Infinity 11, 14
Interfacebibliothek 17f, 23

Konvertierung 19

NaN 12, 78
Nachnull 27
Null 11

O-Exception 26
O-Exception, maskiert 26
O-Exception, unmaskiert 26
Operationscode 21, 36
Overflow 15, 26, 30

P-Exception 26, 31
Parameterfeld 19
Prozessor 8, 18
Prozessorstatuswort 31
Pseudonull 11

Rundung 14

Statuswort 12, 21
Steuerwort 13, 31

U-Exception 26, 31
Underflow 15, 26, 31
Unnormal 11

Vorzeichen 23f

WM87-Emulator 8

zerodivide 15